EXTENDED TRUST-TECH METHODOLOGY FOR NONLINEAR OPTIMIZATION: ANALYSES, METHODS AND APPLICATIONS

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Bin Wang

January 2012

EXTENDED TRUST-TECH METHODOLOGY FOR NONLINEAR OPTIMIZATION: ANALYSES, METHODS AND APPLICATIONS

Bin Wang, Ph.D.

Cornell University 2012

Many theoretical and practical problems can be formulated as a global optimization problem. Traditional local optimization methods can only attain a local optimal solution and be entrapped in the local optimal solution; while existing global optimization algorithms usually sparsely approximates the global optimal solution in a stochastic manner. In contrast, the *transformation under stability-retaining equilibrium characterization (TRUST-TECH)* methodology prevails over existing algorithms due to its capability of locating multiple, if not all, local optimal solutions to the optimization problem deterministically and systematically in a tier-by-tier manner. The TRUST-TECH methodology was developed to solve unconstrained and constrained nonlinear optimization problems. This work extends the TRUST-TECH methodology by incorporating new analytical results, developing new solution methods and solving new problems in practical applications.

This work first provides analytical results regarding the invariance of partial stability region in quasi-gradient systems. Our motivation is to resolve numerical difficulties arising in implementations of trajectory based methods, including TRUST-TECH. Improved algorithms were developed to resolve these issues by altering the original problem to speed-up movement of the trajectory. However, such operations can lead the trajectory converge to a different solution, which could be undesired under specific situations. This work attempts

to answer the question regarding invariant convergence for a special class of numerical operations whose dynamical behaviours can be characterized by a quasi-gradient dynamical system. To this end, we study relationship between a gradient dynamical system and its associated quasi-gradient system and reveal the invariance of partial stability region in the quasi-gradient system. These analytical results lead to methods for checking invariant convergence of the trajectory starting from a given point in the quasi-gradient system and the algorithm to maintain invariant convergence.

This work also develops new solution methods to enhance TRUST-TECH's capability of solving constrained nonlinear optimization problems and applies them to solve practical problems arising in different applications. Specifically, TRUST-TECH based methods are first developed for feasibility computation and restoration and are applied to power system applications, including power flow computation and feasibility restoration for infeasible optimal power flow problems. Indeed, a unified framework based on TRUST-TECH is introduced for analysing feasibility and infeasibility for nonlinear problems. Secondly, the *TRUST-TECH based interior point method (TT-IPM)* and the *reduced projected gradient method* are developed to better tackle constrained nonlinear optimization problems. As application, the TT-IPM method is used to solve mixed-integer nonlinear programs (MINLPs). Finally, this work develops the *ensemble of optimal, input-pruned neural networks using TRUST-TECH (ELITE)* method for constructing high-quality neural network ensembles and applies ELITE to build a short-term load forecaster named *ELITE-STLF* with promising performance.

Possible extensions of the TRUST-TECH methodology to a much broader range of optimization models, including multi-objective optimization and variational optimization, are suggested for future research efforts.

BIOGRAPHICAL SKETCH

Bin Wang was born in a mountain village in Yiyang, Jiangxi Province, China on October 29, 1979. He received the B.E. degree from Tong Ji University, Shanghai, China in 2000 and the M.S. degree from Jiao Tong University, Shanghai, China in 2003, both in Electrical Engineering. After working in Chem-Tech Corp., Shanghai, China for three years as a chief software engineer in developing a medical image based computer-aided diagnosis (CAD) system for lung cancers, he moved to the United States to pursue his higher education. From 2006 to 2007, he was a Ph.D. student in Computational Bioinformatics and Bio-imaging Laboratory (CBIL), Virginia Tech, working on medical image analysis and machine learning methods for bioinformatics. In August 2007, he transferred to the School of Electrical and Computer Engineering at Cornell University, to pursue his doctoral degree. His primary research interests are in nonlinear systems theory and global optimization methods and their applications to different areas, including machine learning, power systems and computer vision.

Dedicated to my family.

ACKNOWLEDGEMENTS

This thesis would not have been possible without the guidance and the help of several individuals who in one way or another contributed and extended their valuable assistance in the preparation and completion of this work.

First and foremost, I would like to express my sincerest gratitude to my advisor Prof. Hsiao-Dong Chiang for the continuous support of my Ph.D study and research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Ph.D study.

I wish to express my warm and sincere thanks to Prof. Anthony Reeves and Prof. Gene Hwang for serving on my thesis committee and for their encouragement, insightful comments, and hard questions.

My life at Cornell would have be much more difficult without the help of members in my research group and fellow students at the School of Electrical and Computer Engineering. Especially, I would like to appreciate Dr. Hua Li, Pat, Simone, Warut, Cheng, and Stephen for their support and friendship. I also had enjoyable discussions with Dr. Dong-Yuan Shi, Dr. Li-Xia Cao and Dr. Quan-Yuan Jiang for sharing their experiences of research and knowledge in power systems.

I wish to thank my friends Yi-Tan Zhu, Yuan-Jian Feng, Iris Yu and Charlene Chen for helping me get through the difficult times, and for all the emotional support, entertainment, and caring they provided.

Last but no the least, I would like to express my deepest love to my family: my parents Lan-Gen Wang and He-Lian Wang, and my brothers Bo and Tao. I couldn't have gone so far without their endless support and encouragement.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Nonlinear Optimization

Many theoretical and practical problems can be formulated as the following global optimization problem:

$$\min_{x \in K} \ f(x), \quad f : R^n \to R, \ K \subseteq R^n, \tag{1.1}$$

where, K is the domain over which the minimum of the objective function $f(x)$ is to be sought. Because of the nonlinearity of $f(x)$ and constraints which define K, real world problems usually contains many local optimal solutions. Obtaining a global optimal solution to (1.1) is of primary importance in real applications and is a very challenging problem.

There has been a wealth of research efforts focused on developing effective and robust methods to solve the optimization problem (1.1). Existing optimization methods for solving (1.1) can be roughly categorized into two types. The first type is called local methods, such as Newton's method, the quasi-Newton method and the conjugate gradient method. These methods usually solve first-order necessary conditions numerically to find local optimal solutions to (1.1). They are generally deterministic and fast to compute a local optimal solution, but can be entrapped in the local optimal solution. The other type is called global methods, such as genetic algorithms, particle swarm optimization and simulated annealing. These methods generally use stochastic mechanisms to escape from a local optimal solution and directly search for an approximation to the global optimal solution to (1.1). Global methods are good at locating

promising areas, but they are generally computationally demanding to find a good approximation to the global optimal solution. It is desirable to develop a deterministic method that can not only escape from a local optimal solution, but compute multiple local optimal solutions to the optimization problem (1.1).

1.2 TRUST-TECH Methodology

1.2.1 The methodology

Recently, a methodology called *transformation under stability-retaining equilibrium characterization (TRUST-TECH)*, has been developed to solve the global optimization problem (1.1) [30, 32, 92, 101, 104]. It has been successfully applied to solve machine learning problems including optimally training ANNs [37] and estimating optimal parameters for finite mixture models [143], and to solve the optimal power flow problem [38].

TRUST-TECH solves the optimization problem (1.1) by first looking for a dynamical system such that the stable equilibrium points (SEPs) in the dynamical system have one-to-one correspondence with local optimal solutions to the optimization problem. Because of such correspondence, the problem of computing multiple local optimal solutions to the optimization problem is then transformed to finding multiple stability regions in the dynamical system, each of which contains a distinct SEP. An SEP can be computed with the trajectory method or using a local method with a trajectory point in its stability region as the initial point [32, 104].

Compared with traditional local methods, TRUST-TECH based methods can escape from a local optimal solution and approach other local optimal solutions in a systematic and deterministic way. TRUST-TECH accomplishes this via two stages: firstly, transforming the study optimization problem into the task of finding the SEPs in a generalized gradient system; and secondly, exploring the stability region of each SEP, which is itself a local optimal solution. Another distinguishing feature of TRUST-TECH is its effective cooperation with existing local and global methods. This cooperation starts with a global method for obtaining promising solutions. Then by working with robust, fast local methods, TRUST-TECH efficiently searches the neighbouring subspace of the promising solutions for new local optimal solutions in a tier-by-tier manner. A high-quality optimal solution, and possibly the global optimal solution, can be found from the multiple local optimal solutions.

1.2.2 Distinction from other trajectory methods

The idea of using dynamical system concepts to solve optimization problems is not new. A good review is provided by Diener [50] for the application of trajectory methods to unconstrained global optimization problems. For equality constrained optimization problems, Yamashita [177] defined a nonlinear autonomous system and developed a trajectory continuation method to find multiple critical points in the system, which are revealed to be local optimal solutions to the optimization problem. This method, however, is only capable of finding multiple local optimal solutions in a single bounded and connected feasible region. In [178], Yang *et al.* combined two trajectory methods to find multiple stationary points of an objective function. Their first method used a

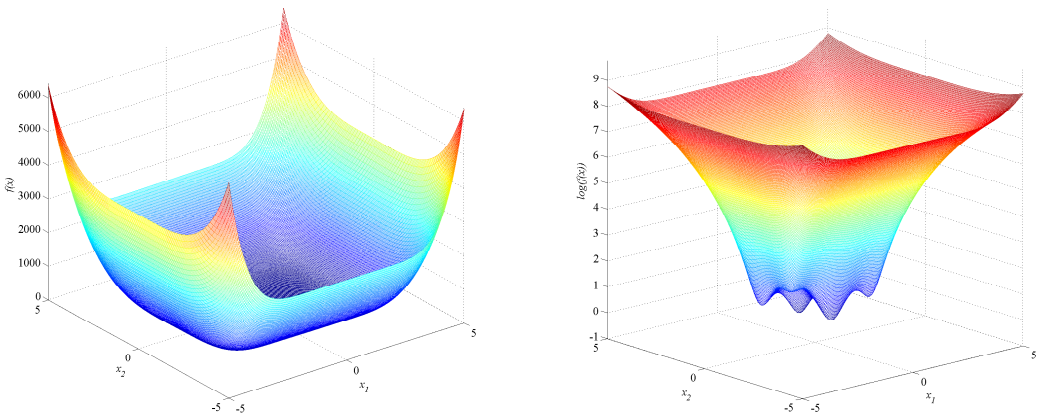
homotopy scheme to trace a trajectory until a stationary point of the objective function and the other method considered a relaxation scheme to connect multiple stationary points with different trajectories.

TRUST-TECH methodology distinguishes itself from existing trajectory methods with following features. Firstly, development of the TRUST-TECH methodology is rooted on the theoretical results for characterizing stability regions in autonomous dynamical systems. Secondly, TRUST-TECH finds another local optimal solution (or feasible component) from a given local optimal solution (or feasible component) by detecting another stability region in the constructed dynamical system. This detection is carried out by detecting the stability boundary between the two stability regions, or more specifically, detecting the decomposition point or the exit point on the stability boundary. Thirdly, the TRUST-TECH methodology is able to find multiple disjoint (path-connected) feasible regions and to compute multiple local optimal solutions within each feasible region. Finally, existing local solvers can be effectively incorporated into TRUST-TECH methodology for efficiently computing local optimal solutions in the identified stability regions.

1.2.3 An illustrative example

We now give a simple example to illustrate how TRUST-TECH solves a global optimization problem. The unconstrained problem to be optimized is a polynomial as follows:

$$\begin{array}{rl} \min_{x \in R^2} f(x) = & 4x_1^2 - 2.1x_1^4 + \dfrac{x_1^6}{3} + x_1x_2 - 4x_2^2 + 4x_2^4 \\ & -5 \le x_1, x_2 \le 5 \end{array} . \tag{1.2}$$



(a) Plot of $f(x)$ (b) Plot of $\log f(x)$

Figure 1.1: Plots of the objective surface. This objective function possesses six local optimal solutions, all are located in a flat basin.

The objective surface for this problem is shown in Fig. 1.1(a). It can be seen that this objective function possesses a flat basin, within which all the local optimal solutions are located. For a better view of the details, a plot of the objective logarithm is shown in Fig. 1.1(b).

To solve this optimization problem using TRUST-TECH, a negative gradient dynamical system is constructed as follows:

$$\begin{aligned} \dot{x}_1 &= -\frac{\partial f}{\partial x_1} &= -8x_1 + 8.4x_1^3 - 2x_1^5 - x_2 \\ \dot{x}_2 &= -\frac{\partial f}{\partial x_2} &= -x_1 + 8x_2 - 16x_2^3 \end{aligned} \quad . \tag{1.3}$$

There are fifteen equilibrium points in the gradient system, which are listed in Table 1.1. Of these equilibrium points, seven are of type-1 (Jacobian matrix has one eigenvalue with positive real part) and two of type-2 (Jacobian matrix has two eigenvalues with positive real part). The remained six equilibrium points are stable (type-0, all Jacobian eigenvalues have negative real part), which correspond to six local optimal solutions to the optimization problem (1.2). Relation-

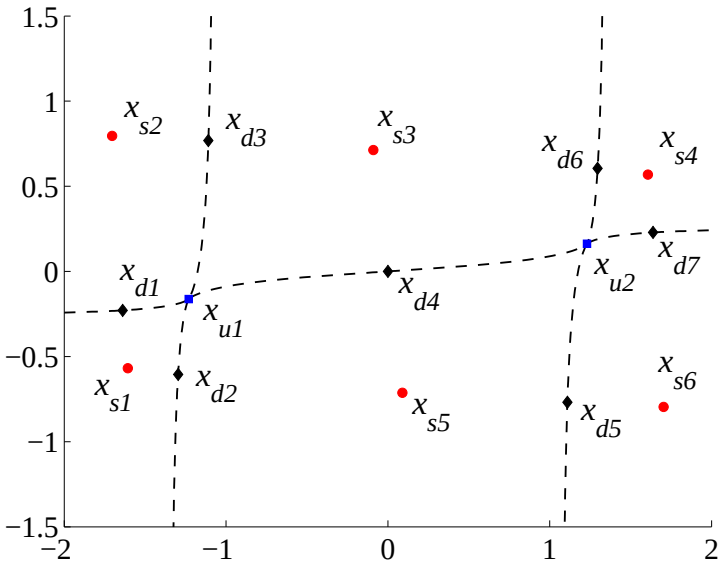


Figure 1.2: Equilibrium points and stability boundaries of (1.3). There are seven type-1 equilibrium points, two type-2 equilibrium points, and six stable equilibrium points corresponding to six local optimal solutions.

ship between equilibrium points and stability boundaries in the gradient system is illustrated in Fig. 1.2. In this figure, stable equilibrium points are of subscript s and highlighted with red dots, type-1 equilibrium points are of subscript d and highlighted with black diamonds, while type-2 equilibrium points are of subscript u and highlighted with blue squares. The dashed curves represent the stability boundaries of the gradient system.

In solving this problem using TRUST-TECH, the limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) method is used as the local solver. The initial point is $(0.0, 0.0)$. Thirty orthogonal directions are generated and used by TRUST-TECH for searching next tier SEPs (local optimal solutions) starting from each found SEP (local optimal solution).

Table 1.1: Equilibrium points in the gradient system

EP	x		$f(x)$	Type
1	-1.6071	-0.5687	2.1043	0
2	-1.6381	-0.2287	2.2294	1
3	-1.7036	0.7961	-0.2155	0
4	-1.2961	-0.6051	2.2295	1
5	-1.2302	-0.1623	2.4963	2
6	-0.0898	0.7127	-1.0316	0
7	-1.1092	0.7683	0.5437	1
8	1.6071	0.5687	2.1043	0
9	0.0898	-0.7127	-1.0316	0
10	1.7036	-0.7961	-0.2155	0
11	0.0000	0.0000	0.0000	1
12	1.1092	-0.7683	0.5437	1
13	1.2961	0.6051	2.2295	1
14	1.6381	0.2287	2.2294	1
15	1.2302	0.1623	2.4963	2

Illustrated in Table 1.2 and Fig. 1.3 to Fig. 1.7, the TRUST-TECH search process can be described as the follows:

Tier 0) Using the specified initial point $x_{01} = (0.0, 0.0)$, the L-BFGS method is carried out and the first local optimal solution x_{s1} is found, which is identical to the initial point since the objective gradient vanishes at it. It needs to note that x_{s1} is NOT a true local optimal solution. In fact, as illustrated in Fig. 1.3, it is a type-1 equilibrium point of the associated gradient system, which corresponds

Table 1.2: The process of TRUST-TECH search to find all local optimal solutions to the test problem.

ID	Search direction	Exit point	Initial point	Local optimal solution	Tier
1	-	-	(0.000, 0.000)	(0.0000, 0.0000)	0
2	(0.918, −0.396)	(−1.249, −0.538)	(−1.311, −0.565)	(−1.6071, −0.5686)	1
3	(−0.929, 0.369)	(−0.993, 0.394)	(−1.042, 0.414)	(−0.0898, 0.7127)	1
4	(−0.918, 0.396)	(−1.230, −0.484)	(−1.211, 0.216)	(0.0898, −0.7127)	2
5	(−0.040, 0.999)	(−1.620, −0.227)	(−1.622, −0.210)	(−1.7036, −0.7961)	2
6	(0.806, 0.592)	(1.246, −0.138)	(1.304, 0.180)	(1.7036, −0.7961)	3
7	(−0.040, −0.999)	(1.662, 0.231)	(1.660, 0.282)	(1.6071, 0.5687)	4

to a saddle point on the objective surface of (1.2).

Tier 1) Starting from x_{s1}, TRUST-TECH searches exit points near the stability boundary in the specified directions. Among these directions, two exit points are located in two directions. The first exit point $(-1.249, -0.538)$ is found along the search direction $(0.918, -0.396)$, and the second exit point $(-0.993, 0.394)$ is found along the search direction $(-0.929, 0.369)$. Two new initial points $x_{02} = (-1.311, -0.565)$ and $x_{03} = (-1.042, 0.414)$ are generated using these two exit points, respectively. The L-BFGS method is applied starting from these initial points and two tier-1 local optimal solutions are found, which are $x_{s2} = (-1.6071, -0.5686)$ and $x_{s3} = (-0.0898, 0.7127)$, respectively, as illustrated in Fig. 1.4.

Tier 2) Starting from x_{s2}, TRUST-TECH finds two exit points $(-1.230, -0.484)$ and $(-1.620, -0.227)$ near the stability boundary along search directions $(-0.918, 0.396)$ and $(-0.040, 0.999)$, respectively. New initial points $x_{04} =$

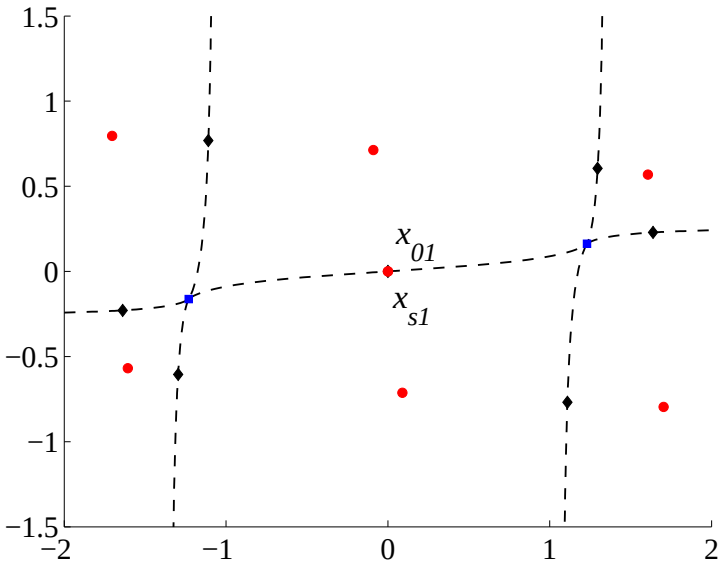


Figure 1.3: Tier 0 of the TRUST-TECH search. The initial SEP $x_{s1} = (0.0, 0, 0)$ from the initial point $x_{01} = (0.0, 0, 0)$.

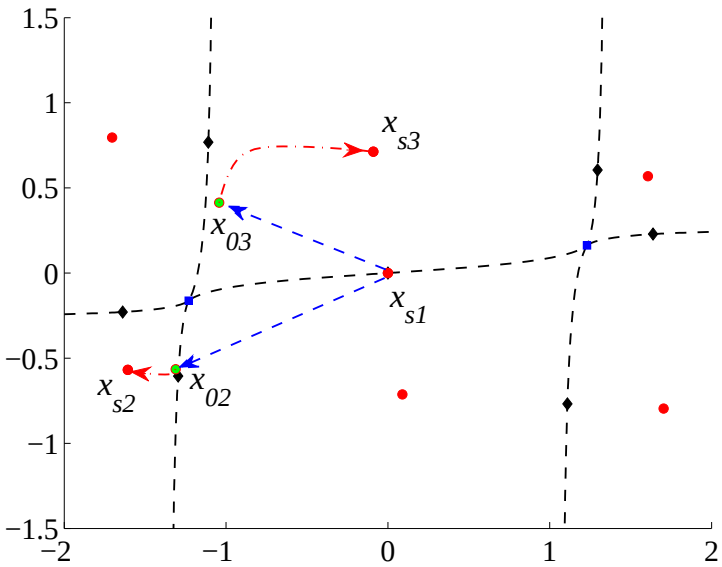


Figure 1.4: Tier 1 of the TRUST-TECH search. Two tier-1 SEPs, $x_{s2} = (-1.6071, -0.5686)$ and $x_{s3} = (-0.0898, 0.7127)$ are found.

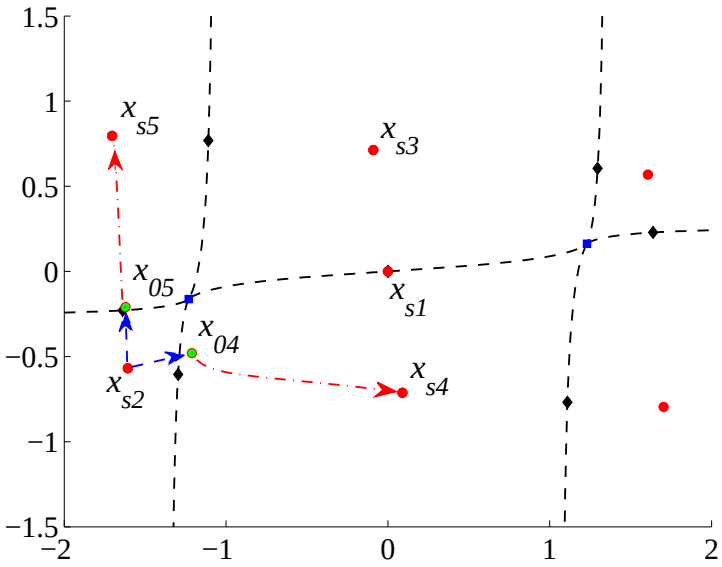


Figure 1.5: Tier 2 of the TRUST-TECH search. Two tier-2 SEPs, $x_{s4} = (0.0898, -0.7127)$ and $x_{s5} = (-1.7036, -0.7961)$ are found starting from the tier-1 SEP x_{s2}.

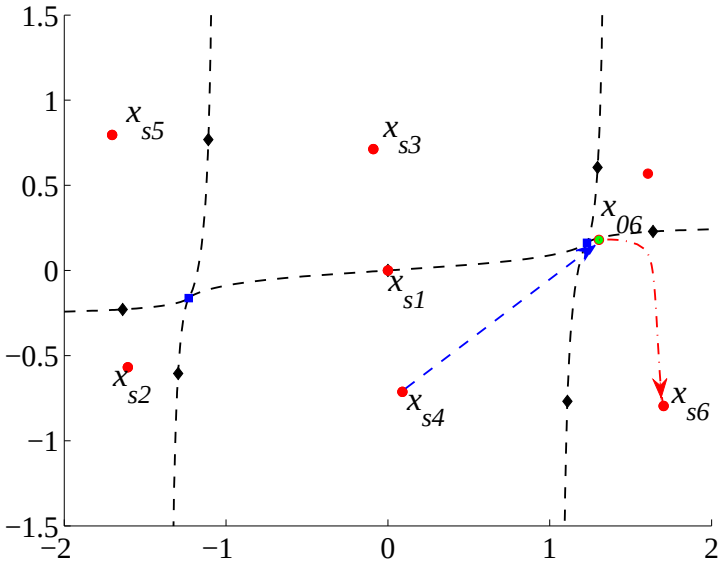


Figure 1.6: Tier 3 of the TRUST-TECH search. A tier-3 SEP, $x_{s6} = (1.7036, -0.7961)$ is found starting from the tier-2 SEP x_{s4}.

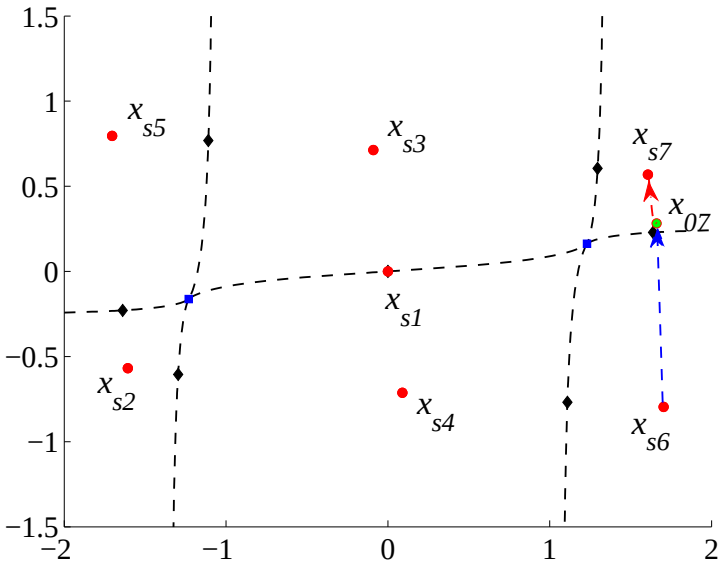


Figure 1.7: Tier 4 of the TRUST-TECH search. A tier-4 SEP, $x_{s7} = (1.6071, 0.5687)$ is found starting from the tier-2 SEP x_{s6}.

$(-1.211, 0.216)$ and $x_{05} = (-1.622, -0.210)$ are generated using the found exit points. The L-BFGS method is applied starting from this initial points and finds tier-2 local optimal solutions $x_{s4} = (0.0898, -0.7127)$ and $x_{s5} = (-1.7036, -0.7961)$, respectively, as illustrated in Fig. 1.5. Starting from x_{s3}, TRUST-TECH finds no new next-tier local optimal solutions in the specified search directions.

Tier 3) Starting from x_{s4}, TRUST-TECH finds an exit points $(1.246, -0.138)$ near the stability boundary along the search direction $(0.806, 0.592)$. A new initial point $x_{06} = (1.304, 0.180)$ is generated using the found exit point. The L-BFGS method is applied starting from this initial points and finds a tier-3 local optimal solutions $x_{s6} = (1.7036, -0.7961)$, as illustrated in Fig. 1.6. Starting from x_{s5}, TRUST-TECH finds no new next-tier local optimal solutions in the specified search directions.

Tier 4) Starting from x_{s6}, TRUST-TECH finds an exit points $(1.662, 0.231)$ near

the stability boundary along the search direction $(-0.040, -0.999)$. A new initial point $x_{07} = (1.660, 0.282)$ is generated using the found exit point. The L-BFGS method is applied starting from this initial points and finds a tier-4 local optimal solutions $x_{s7} = (1.6071, 0.5687)$, as illustrated in Fig. 1.7.

After the above four tiers of search, TRUST-TECH has successfully found all six local optimal solutions to the optimization problem (1.2). Within these local optimal solutions, the global optimal solution is readily available.

1.3 Contributions of This Thesis

This thesis work contributes to the development of TRUST-TECH methodology and general nonlinear programming in the following aspects:

1) Analyze the invariance of partial stability region in quasi-gradient systems which can be used to characterize the dynamical behaviour for a class of numerical methods. Propose two sufficient conditions and corresponding numerical algorithms, in terms of the closest and the controlling unstable equilibrium points (UEPs), respectively, for checking invariant convergence in quasi-gradient systems.
2) Develop the TRUST-TECH based methods for feasibility computation and restoration, which indeed introduces a unified framework for analyzing feasibility and infeasibility of nonlinear problems.
3) Develop the *TRUST-TECH based interior point method (TT-IPM)* for nonlinear constrained global optimization with an improved convergence and with the ability to find multiple local optimal solutions.

4) Develop the *ensemble of optimal, input-pruned neural networks using TRUST-TECH (ELITE)* method for learning high-quality neural network ensembles.

5) Apply the TRUST-TECH based feasibility analysis methods to perform power flow computations with improved convergence and to restore feasibility for infeasible optimal power flow problems.

6) Apply the TT-IPM based method with sensitivity analysis to solve mixed-integer nonlinear programs (MINLP).

7) Develop the ELITE based method, called ELITE-STLF, for power system short-term load forecasting (STLF) with promising performance.

1.4 Organization of This Thesis

Fig. 1.8 shows the organization chart of this thesis. There are three major parts in this thesis, that is, 1) analysis, 2) methods, and 3) applications. The analysis part consists of solely Chapter 2, where theoretical results are derived to describe invariance of partial stability region in quasi-gradient systems and algorithms are proposed for checking invariant convergence for a class of numerical methods which can be characterized as quasi-gradient systems. The methods part consists of Chapter 3, 4 and 6. Specifically, TRUST-TECH based methods are developed for feasibility analysis, which is presented in Chapter 3. In Chapter 4, the TRUST-TECH based interior point method (TT-IPM) is proposed for solving nonlinear constrained global optimization problems. In Chapter 6, the ELITE method is developed for learning high-quality neural network ensembles. The applications part consists of Chapter 3, 5, and 7. Specifically, in Section 3.4,

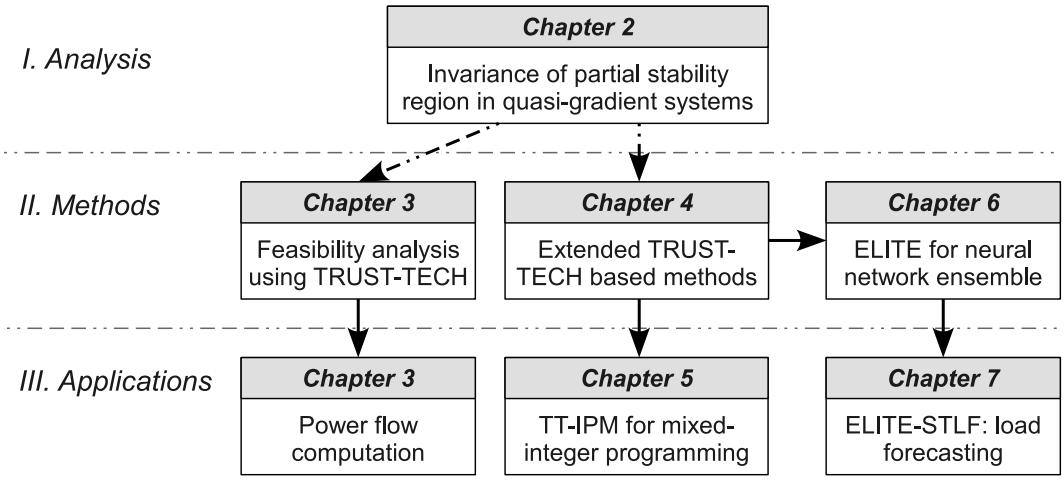


Figure 1.8: Organization chart of this thesis. The three major parts of this thesis, including analysis, methods and applications, are illustrated.

the feasibility analysis methods are used to solve power system problems, including power flow computations and feasibility restoration of optimal power flows. In Chapter 5, the TT-IPM method is applied to solve mixed-integer nonlinear programs. Chapter 7 applies the ELITE method to construct a short-term load forecaster, called ELITE-STLF, with promising results. Finally, the thesis is concluded in Chapter 8 with proposals for a few future research directions.

CHAPTER 2

INVARIANCE OF PARTIAL STABILITY REGION IN QUASI-GRADIENT SYSTEMS

2.1 Introduction

Let us consider the following global optimization problem:

$$\min_{x=(x_1,x_2,\cdots,x_n)\in R^n} f(x), \quad f: R^n \to R, \ f \in C^2. \tag{2.1}$$

The objective function $f(x)$ is generally nonlinear and can have multiple local optimal solutions, each of which satisfies the first-order necessary conditions

$$\nabla f(x) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \cdots, \frac{\partial f}{\partial x_n}\right)^T = 0. \tag{2.2}$$

There have been a lot of research efforts focused on developing effective and robust methods to solve (2.1). Existing optimization methods for solving (2.1) can be roughly categorized into two types. The first type is called local methods, such as Newton's method, quasi-Newton method, and conjugate gradient method. These methods attempt to solve (2.2) numerically to find local optimal solutions to (2.1). They are generally deterministic and fast to compute a local optimal solution, but can be entrapped in the local optimal solution. The other type is called global methods, such as genetic algorithms, particle swarm optimization, and simulated annealing. These methods generally use some stochastic mechanism to escape from a local optimal solution and directly search for an approximation to the global optimum of (2.1). Global methods are good at locating promising areas, but they are generally computationally demanding in finding a high-quality local optimal solution. It is desirable to develop a deterministic method that can not only escape from a local optimal solution, but

compute multiple local optimal solutions to (2.1). Compared with traditional deterministic local methods, TRUST-TECH based methods can escape from a local optimal solution and search multiple local optimal solutions in a systematic and deterministic way. A high-quality solution, and possibly the global optimal solution, can be found from these local optimal solutions.

In practical applications, however, trajectory based methods, including TRUST-TECH, may encounter numerical difficulties when the provided initial point is far away from a solution to the optimization problem (2.1). Specifically, movement of the trajectory can be very slow which results in inefficiency or failure of these methods in finding a local optimal solution to (2.1). Improved algorithms have been developed to address this issue, such as the widely used Levenberg-Marquardt algorithm [66] and the relatively recent pseudo-transient continuation method proposed in [46, 93]. The latter method can be treated as a generalization of the former one. Essentially, these methods try to attack aforementioned numerical issues by altering the original problem to speed-up movement of the trajectory while keeping the solutions untouched. One way to speed-up the trajectory is to multiply the original system with a positive-definite symmetrical matrix (which may vary along the trajectory). However, such operations can lead the trajectory converge to a different solution, which could be undesired under some situations. Therefore, a question raised naturally is that under what conditions the trajectory will converge to a same solution, which is independent of the imposed operations?

This chapter attempts to answer this question regarding invariant convergence via analyzing the invariance of partial stability region in quasi-gradient systems. We first study relationship of the gradient dynamical system $\dot{x} =$

$-\nabla f(x)$ and the associated quasi-gradient system $\dot{x} = -Q\nabla f(x)$, where $Q \in S^N_{++}$ can be any positive-definite symmetrical matrix. Specifically, under a few mild assumptions, the quasi-gradient system possesses several interesting invariant properties. Firstly, it is shown that the quasi-gradient system possesses the same set of equilibrium points with same dynamical properties. Secondly, it is revealed that the stability boundary of each *stable equilibrium point (SEP)* contains the same set of *unstable equilibrium points (UEPs)* in the quasi-gradient system. Thirdly, the closest UEP with respect to an SEP is proved to be invariant. Finally, it is found that the controlling UEP with respect to a search path is also invariant. Based on these invariant properties, the partial stability region of an SEP that is invariant under the change of Q will be optimally approximated with different schemes. Specifically, this invariant partial stability region is first approximated using the energy function level set of the closest UEP. Then, the energy function level set of the controlling UEP is used as another approximation with decreased conservativeness. These lead to methods for checking and preserving invariant convergence of the trajectory starting from a given point in the quasi-gradient system. Numerical simulation and results are presented to support these claims.

2.2 Preliminaries

In this chapter, we assume that the objective function $f(x)$ in (2.1) satisfies the following conditions:

- $f(x)$ is twice continuously differentiable, that is, $f \in C^2 : R^n \to R$;
- $\nabla f(x)$ is Lipschitz continuous, that is, there exists a constant K such that

$|\nabla f(x_1) - \nabla f(x_2)| < K|x_1 - x_2|, \forall x_1, x_2 \in R^n$; and

- $f(x)$ is lower bounded, that is, there exists a constant U such that $f(x) > U$, $\forall x \in R^n$. In other words, the optimization problem (2.1) is solvable.

TRUST-TECH based methods solve the global optimization problem (2.1) by first defining a dynamical system such that SEPs in the dynamical system have one-to-one correspondence with local optimal solutions of the optimization problem (2.1). Because of such correspondence, the problem of computing multiple local optimal solutions of the optimization problem is then transformed to finding multiple stability regions in the defined dynamical system, each of which contains a distinct SEP. An SEP can be computed with a trajectory based method or using a local method with a trajectory point in its stability region being the initial point [32, 104]. For the global optimization problem (2.1), we can define such a desired dynamical system as the following negative gradient system:

$$\dot{x}(t) = -\nabla f(x) = -\left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \cdots, \frac{\partial f}{\partial x_n}\right)^T. \tag{2.3}$$

The solution curve of system (2.3) starting from x_0 at $t = 0$ is called the *trajectory* starting from x_0 and is denoted by $\phi(x_0, t) : R \to R^n$. A state vector $\hat{x}$ is called an *equilibrium point (EP)* of the dynamical system (2.3) if $\nabla f(\hat{x}) = 0$. An equilibrium point $\hat{x}$ is said to be hyperbolic if its Jacobian matrix $-\nabla^2 f(\hat{x})$, with

$$\nabla^2 f(x) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & d\frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}, \tag{2.4}$$

has no eigenvalue with zero real part. A hyperbolic equilibrium point x_s is called a (asymptotically) stable equilibrium point if all the eigenvalues of its Jacobian matrix have negative real part. An equilibrium point is called type-k if its Jacobian matrix has exactly k eigenvalues with positive real part. The stable and unstable manifolds of an equilibrium point $\hat{x}$, denoted as $W^s(\hat{x})$ and $W^u(\hat{x})$, are defined as follows:

$$\begin{aligned} W^s(\hat{x}) &= \{x \in R^n : \phi(x,t) \to \hat{x},\ as\ t \to +\infty\} \\ W^u(\hat{x}) &= \{x \in R^n : \phi(x,t) \to \hat{x},\ as\ t \to -\infty\} \end{aligned} . \tag{2.5}$$

In other words, every trajectory in the stable manifold $W^s(\hat{x})$ converges to $\hat{x}$ as time goes to positive infinity, while every trajectory in the unstable manifold $W^u(\hat{x})$ converges to $\hat{x}$ as time goes to negative infinity.

For a stable equilibrium point, it can be shown that there exists a number $\delta > 0$ such that $\|x - \hat{x}\| < \delta$ implies $\phi(x,t) \to \hat{x}$ as $t \to +\infty$. There are many physical systems containing multiple SEPs. A useful concept for this kind of systems is that of the *stability region* (also called the region of attraction). The stability region of an SEP x_s is defined as:

$$A(x_s) := \{x \in R^n : \lim_{t \to +\infty} \phi(x,t) = x_s\}. \tag{2.6}$$

From a topological point of view, the stability region $A(x_s)$ is an open, invariant and connected set. The boundary of stability region $A(x_s)$ is called the *stability boundary* of x_s and is denoted by $\partial A(x_s)$.

We next introduce several geometrical and topological properties of the gradient system (2.3) which will be useful in this chapter.

Theorem 2.2.1 *[32] If there exist ϵ and δ such that $\|\nabla f(x)\| < \epsilon$ unless $x \in B_\delta(\hat{x})$, where $\hat{x} \in E = \{x | \nabla f(x) = 0\}$ is an equilibrium point, then the gradient system (2.3) is*

completely stable and $f(x)$ is an associated Lyapunov function.

Theorem 2.2.2 *[35] Suppose that all the equilibrium points of the gradient system (2.3) are hyperbolic. Let $x_i, i = 1, 2, \cdots$ be the equilibrium points on the stability boundary $\partial A(x_s)$ of an SEP x_s. Then, the stability boundary is contained in the union of the stable manifolds of the equilibrium points on the stability boundary; in other words,*

$$\partial A(x_s) \subseteq \bigcup_{x_i \in E \cap \partial A(x_s)} W^s(x_s). \tag{2.7}$$

Theorem 2.2.3 *[32] If $\hat{x}$ is a hyperbolic equilibrium point of gradient system (2.3), then 1) $\hat{x}$ is an SEP of system (2.3) if and only if $f(x)$ has an isolated minimum at $\hat{x}$; and 2) $\hat{x}$ is a source of system (2.3) if and only if $f(x)$ has an isolated maximum at $\hat{x}$.*

Theorem 2.2.3 characterizes the relationship between the optimal solutions of the global optimization problem (2.1) and the SEPs of the associated gradient system (2.3). Hence, if x_s is an SEP of (2.3), then it is a local optimal solution of the optimization problem (2.1). Conversely, if x_s is a local optimal solution of (2.1), then it is an SEP of the gradient system (2.3).

The TRUST-TECH based exit point method for solving the optimization problem (2.1) can be described as follows.

TRUST-TECH based exit point method [31]:

Step 1) Construct a path moving away from the initial stable equilibrium point and moving toward its practical stability boundary.

Step 2) Identify the exit point at which the constructed path intersects with the practical stability boundary.

Step 3) Generate one point from the exit point which lies inside the stability region of the corresponding stable equilibrium point.

Step 4) Integrate the system trajectory starting from the point generated at Step 3. This trajectory will converge to the corresponding stable equilibrium point (which is another local optimal solution). Alternatively, apply a local optimization method to find the corresponding local optimal solution from a point in the trajectory.

The point generated at Step 3) is generated close to the exit point, thus close to the stability boundary. This point is far away from the corresponding stable equilibrium point. The trajectory integrated from this point can move very slowly which results in inefficiency or failure of TRUST-TECH in finding the corresponding stable equilibrium point. Improved algorithms, such as pseudo-transient continuation method can be used to speed-up the movement of the trajectory. This type of algorithms, described in the simple form, numerically integrate the following altered dynamical system:

$$\dot{x} = -Q\nabla f(x),\ Q \in S^n_{++}, \tag{2.8}$$

where, S^n_{++} denotes the set of $n \times n$ real positive definite (symmetric) matrices. The matrix Q in (2.8) is used to improve the scaling of the problem and can be adaptively changed along the integration process. In this paper, we call the dynamical system described by (2.8) the quasi-gradient system and denote it as $d(Q)$. However, it is a natural concern that such operations can lead the trajectory converge to a different solution, which could be undesired under some situations. Therefore, a question raised is that under what conditions the trajectory will converge to a same solution, which is independent of the imposed matrix Q?

2.3 Invariant Properties of the Quasi-Gradient System

To answer the above question regarding invariant convergence, invariance of partial stability region in the quasi-gradient system will be studied in this chapter. To this end, in this section, we first study relationship between the gradient system $\dot{x} = -\nabla f(x)$ and its associated quasi-gradient system (2.8). We assume that the gradient system (2.3) satisfies the following conditions:

A1) All the equilibrium points on the stability boundary are hyperbolic;

A2) The number of equilibrium points on the stability boundary is finite; and

A3) The stability region for a stable equilibrium point is bounded.

To facilitate analysis in the sequel, we first give definitions of some important terms which will be encountered frequently throughout this chapter.

Definition A continuously differentiable function $V(x) : R^n \rightarrow R$ is called an energy function with respect to the dynamical system (2.3) if it satisfies the following conditions:

- $\dot{V}(\phi(x,t)) \leq 0$ if $x \notin E$ with E being the set of equilibrium points;
- The set $\{t \in R : \dot{V}(\phi(x,t)) = 0\}$ has measure 0 in R, if $x \notin E$; and
- $V(\phi(x,t))$ is bounded implies $\phi(x,t)$ is bounded.

Definition The closest UEP with respect to an SEP x_s is a UEP x_c on the stability boundary $\partial A(x_s)$ which attains the minimum energy function value on $\partial A(x_s)$.

Definition The exit point x_e with respect to a search path $R(\lambda)$ passing through $x_p \in A(x_s)$, where $\lambda \geq 0$ and $R(0) = x_p$, is defined as the first intersection point of the search path with the stability boundary $\partial A(x_s)$.

The intersection of the search path $R(\lambda)$ and the stability boundary $\partial A(x_s)$ is assumed to satisfy the transversality condition.

Definition The controlling UEP with respect to a search path $R(\lambda)$ is a UEP x_c on the stability boundary $\partial A(x_s)$ whose stable manifold contains the exit point x_e.

It it first shown that the quasi-gradient system (2.8) has the same set of equilibrium points with same dynamical properties, which is invariant to the change of the matrix Q.

Theorem 2.3.1 (Invariance of the equilibrium points) *The quasi-gradient system (2.8) has the same equilibrium points and the same inertia as that of the system (2.3). Moreover, all the equilibrium points are hyperbolic.*

Proof It is trivial to show that quasi-gradient systems $d(Q_1)$ and $d(Q_2)$ with $Q_1 \neq Q_2$ have the same set of equilibrium points since they are defined by $F(x) = 0$. Let λ_x denote the eigenvalue of the Jacobian matrix

$$Q_1 J_x = Q_1 \left[\frac{\partial F(x)}{\partial x} \right] \tag{2.9}$$

at the equilibrium point x, and let v be the associated eigenvector, i.e.,

$$Q_1 J_x v = \lambda_x v. \tag{2.10}$$

Since Q1 is a positive definite matrix, we can write . Hence,

$$Q_1^{1/2}(Q_1^{1/2}J_x - \lambda_x Q_1^{-1/2})v = 0, \tag{2.11}$$

or

$$Q_1^{1/2}(Q_1^{1/2}J_x Q_1^{1/2} - \lambda_x I)Q_1^{-1/2}v = 0. \tag{2.12}$$

Because $Q_1^{1/2}$ and $Q_1^{-1/2}$ are non-singular, the eigenvalues of $Q_1^{1/2}J_xQ_1^{1/2}$ and that of Q_1J_x are the same. Moreover, $Q_1^{1/2}J_xQ_1^{1/2}$ is congruent to J_x. From Sylvester's inertial theorem, the congruence transformation preserves the inertia of matrix.

Similarly, it follows that the eigenvalues of $Q_2^{1/2}J_xQ_2^{1/2}$ and Q_2J_x are the same, and $Q_2^{1/2}J_xQ_2^{1/2}$ is congruent to J_x. This theorem follows. ■

It is obvious that the objective function $f(x)$ is an energy function for the gradient system (2.3) constructed in TRUST-TECH methodology. It is a direct consequence of the positive-definiteness of Q that $f(x)$ is also a common energy function for the quasi-gradient system, independent of the choice of Q.

Lemma 2.3.1 *The objective function $f(x)$ is a common energy function for the quasi-gradient system (2.8) for every $Q \in S_{++}^n$.*

Proof To show $f(x)$ is an energy function for the system (2.8), we only need to show it is non-increasing along any trajectory. In fact, since $Q \in S_{++}^n$, in the quasi-gradient system (2.8), we have

$$\dot{V}(x) = \langle \frac{\partial V}{\partial x}, \dot{x} \rangle = \langle \nabla f(x), -Q\nabla f(x) \rangle = -\langle \nabla f(x), Q\nabla f(x) \rangle \leq 0. \tag{2.13}$$

Hence, $V = f(x)$ is an energy function for the quasi-gradient system (2.8). ■

Development of TRUST-TECH methodology is rooted on the theoretical foundation of characterization of the stability boundary in the gradient system (2.3). Moreover, a central task in implementing TRUST-TECH based methods is to detect the stability boundary and then using the information about the stability boundary to facilitate the search of neighbouring stable equilibrium points (local optimal solutions). Theorem 2.2.2 reveals that, for general autonomous dynamical systems, there is a direct relationship between the stability boundary and stable manifolds of the equilibrium points on it.

For the special class of dynamical systems satisfying assumptions A1) through A3), such as the negative gradient system (2.3) constructed in TRUST-TECH methodology, Theorem 2.3.2 shows that structure of the stability boundary can be explicitly defined as the union of stable manifolds of type-1 equilibrium points.

Theorem 2.3.2 (Characterization of the stability boundary) *[32] Let x_s be an SEP of the system (2.8) and x_i, $i = 1, 2, \cdots$ be the type-1 equilibrium points lying on $\partial A_Q(x_s)$. If assumptions A1) to A3) are satisfied, then*

$$\partial A_Q(x_s) = \bigcup_{x_i \in E_1 \cap \partial A_Q(x_s)} \overline{W_Q^s(x_i)}. \tag{2.14}$$

Note that Theorem 2.3.2 also shows if assumptions A1) to A3) are satisfied, both the closest UEP w.r.t. an SEP x_s and the controlling UEP w.r.t. a search path $R(\lambda)$ will generally be type-1 boundary equilibrium points.

Next we will show in Theorem 2.3.3 that the equilibrium points on the stability boundary of each stable equilibrium point will also keep the same in quasi-gradient system (2.8). To facilitate the proof of Theorem 2.3.3, the following two lemmas will be given first.

Lemma 2.3.2 *Suppose* $A, B \in S^n_{++}$, *and* $0 < a_1 \le \lambda(A) \le b_1$, $0 < a_2 \le \lambda(B) \le b_2$, *where* $\lambda(A)$ *denotes the eigenvalues of A. Then,*

$$a_1 a_2 \le \lambda(AB) = \lambda(BA) \le b_1 b_2. \tag{2.15}$$

Proof Since $A \in S^n_{++}$, we have $\lambda(A) \in [a_1, b_1]$ if and only if

$$a_1 \|x\|^2 \le \langle x, Ax \rangle \le b_1 \|x\|^2, \ \forall x \in R^n, \tag{2.16}$$

where $\langle \cdot, \cdot \rangle$ is the inner product. Similarly,

$$a_2 \|x\|^2 \le \langle x, Bx \rangle \le b_2 \|x\|^2, \ \forall x \in R^n. \tag{2.17}$$

Now let $A^{1/2}$ be the positive definite square root of A, then $\forall x \in R^n$,

$$\begin{aligned} \langle x, ABx \rangle &= \langle x, A^{1/2} B A^{1/2} x \rangle = \langle A^{1/2} x, B A^{1/2} x \rangle \\ &\le b_2 \langle A^{1/2} x, A^{1/2} x \rangle = b_2 \langle x, Ax \rangle \le b_1 b_2 \|x\|^2 \end{aligned}, \tag{2.18}$$

where, the second equality holds because $AB = A^{1/2} A^{1/2} B$ and $A^{1/2} B A^{1/2}$ have same eigenvalues. Similarly, we have

$$\langle x, ABx \rangle \ge a_1 a_2, \ \forall x \in R^n. \tag{2.19}$$

Hence, $\lambda(AB) = \lambda(BA) \in [a_1 a_2, b_1 b_2]$. ■

Lemma 2.3.3 *Suppose* x_s *is an SEP of the quasi-gradient system (2.8). Then there exist* $r > 0$ *and* $\epsilon > 0$ *such that* $B_r(x_s) \subset A_Q(x_s)$, *and for every matrix* $\overline{Q}$ *satisfying* $|Q - \overline{Q}| < \epsilon$, $B_r(x_s) \subset A_{\overline{Q}}(x_s)$.

Proof Let $J_{x_s} = \nabla^2 f(x_s)$. Since J_{x_s} is symmetrical, it has only real eigenvalues. Moreover, because x_s is a s.e.p. of $d(Q)$, all eigenvalues of the linearized vector field at x_s are negative. By Lemma 2.3.2,

$$\sigma(-QJ_{x_s}) \ \le \ -\underline{\lambda}_Q \underline{\lambda}_J \ = \ -c, \tag{2.20}$$

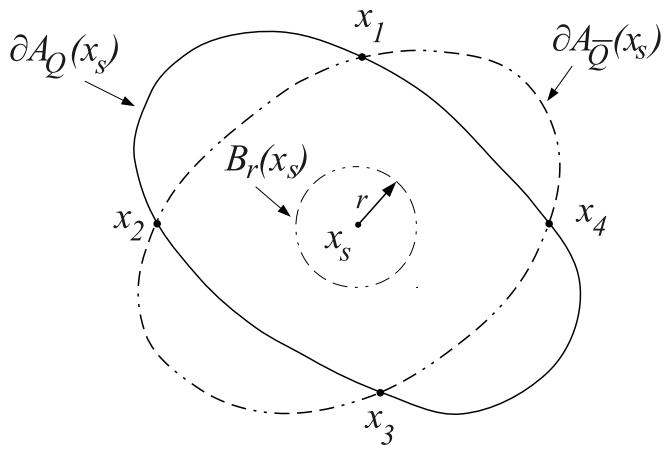


Figure 2.1: Illustration of Lemma 2.3.3 for a common ball inclusion.

where, $\underline{\lambda}_Q$ and $\underline{\lambda}_J$ are the minimum eigenvalues of Q and J_{x_s}, respectively, and $c > 0$. Since

$$\langle -QJ_{x_s}x, x\rangle \ \leq \ -c|x|^2, \ \forall x \in R^n, \tag{2.21}$$

it follows from the definition of the derivative that

$$\lim_{x\to x_s} \frac{|-Q\nabla f(x) + QJ_{x_s}(x-x_s)|}{|x-x_s|} \ = \ 0, \tag{2.22}$$

or by Cauchy inequality

$$\lim_{x\to x_s} \frac{\langle -Q\nabla f(x) + QJ_{x_s}(x-x_s), x-x_s\rangle}{|x-x_s|^2} \ = \ 0. \tag{2.23}$$

Then, for any small number $c_1 > 0$, there exists $r > 0$ such that if $|x - x_s| \leq r$ (or $x \in B_r(x_s)$) implies

$$\langle -Q\nabla f(x) + QJ_{x_s}(x-x_s), x-x_s\rangle \ \leq \ c_1|x-x_s|^2, \tag{2.24}$$

or

$$\begin{aligned} \langle -Q\nabla f(x), x-x_s\rangle \ &\leq \ \langle -QJ_{x_s}(x-x_s), x-x_s\rangle + c_1|x-x_s|^2 \\ &\leq \ -(c-c_1)|x-x_s|^2 \\ &= \ -c_2|x-x_s|^2 \end{aligned} \quad , \tag{2.25}$$

where we choose $c_1 < c$, so $c_2 > 0$.

Therefore, there exists a neighbourhood of Q such that for every matrix $\overline{Q}$ in this neighbourhood we have

$$\langle -\overline{Q}\nabla f(x), x - x_s \rangle \ \leq \ -c_3|x - x_s|^2, \ \forall x \in B_r(x_s), \tag{2.26}$$

where $c_3 > 0$ and $\|Q - \overline{Q}\| < \epsilon$.

Let $V(x) = \langle x - x_s, x - x_s \rangle$. Then, the derivative of V along the trajectory of $\dot{x} = -\overline{Q}\nabla f(x)$ is

$$\dot{V}_{\overline{Q}}(x) \ = \ 2\langle -\overline{Q}\nabla f(x), x - x_s \rangle \ \leq \ 0, \ \forall x \in B_r(x_s). \tag{2.27}$$

Now, consider the set $S(r) = \{x \in R^n : V(x) \leq r^2\}$. It is clear that the set $S(r)$ is bounded for any bounded r. From (2.27), it follows that $S(r)$ is an invariant set of $d(\overline{Q})$. Also it can be shown from (2.27) that x_s is the largest invariant set of $d(\overline{Q})$ contained in $S(r)$. Consequently, all the trajectories of $\dot{x} = -\overline{Q}\nabla f(x)$ starting from $x \in S(r)$ will approach x_s as $t \to \infty$. Thus, $B_r \subseteq A_{\overline{Q}}(x_s)$. ■

Theorem 2.3.3 (Local invariance of boundary equilibrium points) *Suppose x_s is an SEP of $d(Q)$. If the system (2.8) satisfies assumptions (A1) to (A3), then there exists $\epsilon > 0$ such that for every $\overline{Q}$ satisfying $\|\overline{Q} - Q\| < \epsilon$, the stability boundaries $\partial A_Q(x_s)$ and $\partial A_{\overline{Q}}(x_s)$ have the same set of type-1 equilibrium points.*

Proof Let $U \subset R^n$ be a neighbourhood of the e.p. x_i of $d(Q)$ and define the local stable and unstable manifolds of $d(Q)$ as

$$\hat{W}^s_Q(x_i) = \{x \in U | \phi_{t\to+\infty}(x; Q) \to x_i, \text{ and } \phi_t(x; Q) \in U, \ \forall t \geq 0\}$$
$$\hat{W}^u_Q(x_i) = \{x \in U | \phi_{t\to-\infty}(x; Q) \to x_i, \text{ and } \phi_t(x; Q) \in U, \ \forall t \leq 0\}$$

By Theorem 2-3 of [40], for each UEP x_i on the stability boundary $\partial A(Q)$ and for any $r > 0$ such that $B_r(x_s) \subseteq A_Q(x_s)$, we have $\hat{W}^u_Q(x_i) \cap B_r(x_s) \neq \emptyset$, $i = 1, 2, \cdots, N$. In particular, according to Lemma 2.3.3, there exist $r > 0$ and $\hat{\epsilon}_1 > 0$ such that $B_r(x_s) \subseteq A_{\overline{Q}}(x_s)$, for every $\overline{Q}$ satisfying $\|\overline{Q} - Q\| < \hat{\epsilon}_1$.

Considering the local unstable and stable manifolds depend continuously on parameter Q [79], we have that, for each $x_i \in E \cap \partial A(Q)$, there exists an $\epsilon_i > 0$ such that for every matrix Q_i satisfying $\|Q_i - Q\| < \epsilon_i$ implies

$$\hat{W}^u_{Q_i}(x_i) \cap B_r(x_s) \neq \emptyset. \tag{2.28}$$

Let $\hat{\epsilon}_2 = \min\{\epsilon_1, \cdots, \epsilon_N\}$. Then for every $\hat{Q}$ satisfying $\|\hat{Q} - Q\| < \hat{\epsilon}_2$, we have

$$\hat{W}^u_{\hat{Q}}(x_i) \cap B_r(x_s) \neq \emptyset, \text{ for } i = 1, 2, \cdots, N. \tag{2.29}$$

Now take $\hat{\epsilon} = \min\{\hat{\epsilon}_1, \hat{\epsilon}_2\}$. Then for every $\overline{Q}$ satisfying $\|\overline{Q} - Q\| < \hat{\epsilon}$, we have

$$\hat{W}^u_{\overline{Q}}(x_i) \cap B_r(x_s) \neq \emptyset, \text{ for } i = 1, 2, \cdots, N. \tag{2.30}$$

In particular, we have

$$\hat{W}^u_{\overline{Q}}(x_i) \cap A_{\overline{Q}}(x_s) \neq \emptyset, \text{ for } i = 1, 2, \cdots, N. \tag{2.31}$$

Hence, $x_1, \cdots, x_N$ are on the stability boundary $\partial A_{\overline{Q}}(x_s)$.

To complete the proof, we need to show that no new type-1 equilibrium points can be introduced to $\partial A_{\overline{Q}}(x_s)$. Note that the unstable manifolds of a type-1 equilibrium point consist of two 1-dimensional solution curves of the system (2.8), each of which connects this type-1 equilibrium point with a stable equilibrium point. Now assume there is a new type-1 equilibrium point x'_e introduced to $\partial A_Q(x_s)$. Considering that the quasi-gradient system (2.8) has the same set of equilibrium points for every $Q \in \mathcal{S}^n_{++}$, x'_e must come from the stability boundary

of anther s.e.p., say x'_s. However, for x'_s, there exists $\epsilon' > 0$, such that for all $\tilde{Q}$ satisfying $\|\tilde{Q} - Q\| < \epsilon'$, the following relation holds

$$E \cap \partial(A_Q(x'_s)) \subseteq E \cap \partial A_{\tilde{Q}}(x'_s),$$

that is, no type-1 equilibrium point can break away from $\partial A_{\tilde{Q}}(x'_s)$. Hence, if we take $\epsilon = \min\{\hat{\epsilon}, \epsilon'\}$, then for all $\overline{Q}$ satisfying $\|\overline{Q} - Q\| < \epsilon$, the set of type-1 equilibrium points will keep the same on $\partial A_{\overline{Q}}(x_s)$. ■

Theorem 2.3.3 shows that boundary equilibrium points are invariant locally in terms of the change of Q. In fact, validity of this invariant property can be extended such that the boundary equilibrium points are invariant globally in dependent of the change of Q. This global invariance is described in Theorem 2.3.4. To facilitate the proof of Theorem 2.3.4, Lemma 2.3.4 is given first to show the path-wise invariance of the boundary equilibrium points.

Lemma 2.3.4 (Path-wise invariance of boundary equilibrium points) *Suppose x_s is an SEP for the system (2.8). Let $q(\lambda) = Q_\lambda : [0,1] \to \mathcal{S}^n_{++}$ be a continuous function with $q(0) = Q_1$ and $q(1) = Q_2$. If $d(Q_\lambda)$ satisfies assumption (A1) to (A3) for all $\lambda \in [0,1]$, then the stability boundaries $\partial A_{Q_1}(x_s)$ and $\partial A_{Q_2}(x_s)$ have the same set of type-1 equilibrium points.*

Proof Since $d(Q_\lambda)$ satisfies assumptions $(A1)$ and $(A2)$, $\forall \lambda \in [0,1]$, it follows from Theorem 2.3.3 that there exist an $\epsilon_\lambda > 0$, such that for every $\tilde{Q}$ satisfying $\|\tilde{Q} - Q_\lambda\| < \epsilon_\lambda$, $\partial A(Q_\lambda)$ and $\partial A(\tilde{Q})$ have the same set of equilibrium points. Hence, $\mathcal{B} = \{B_{\epsilon_\lambda}(Q_\lambda), \lambda \in [0,1]\}$ forms an open cover of the set $\mathcal{Q} = \{Q_\lambda,\ \lambda \in [0,1]\}$. On the other hand, since $\mathcal{Q}$ is a compact set, every open cover of $\mathcal{Q}$ has a finite subcover. Therefore, there exist $\hat{\mathcal{B}} = \{B_{\epsilon_{\lambda_i}}(Q_{\lambda_i}), \lambda_i \in [0,1], i = 1, \cdots, k\}$ being a finite cover of $\mathcal{Q}$, such that $\mathcal{Q} \nsubseteq \hat{\mathcal{B}} - B_{\epsilon_{\lambda_i}}(Q_{\lambda_i})$, $\forall i = 1, \cdots, k$.

In addition, the elements in $\hat{\mathcal{B}}$ can be reordered such that $Q_1 \in B_{\epsilon_{\tilde{\lambda}_1}}$, $Q_2 \in B_{\epsilon_{\tilde{\lambda}_k}}$ and $B_{\epsilon_{\tilde{\lambda}_i}} \cap B_{\epsilon_{\tilde{\lambda}_{i+1}}} \neq \emptyset$, $\forall i = 1, \cdots, k-1$. Let $\hat{Q}_i \in B_{\epsilon_{\tilde{\lambda}_i}} \cap B_{\epsilon_{\tilde{\lambda}_{i+1}}}$, then $\partial A(Q_1)$, $\partial A(\hat{Q}_1), \cdots, \partial A(\hat{Q}_{k-1})$, $\partial A(Q_2)$ all have the same set of type-1 equilibrium points. Specifically, $\partial A(Q_1)$ and $\partial A(Q_2)$ have the same set of type-1 equilibrium points. ■

Theorem 2.3.4 (Global invariance of boundary equilibrium points) *If the quasi-gradient system (2.8) satisfies assumptions A1) through A3), then the stability boundary $\partial A(x_s)$ for an SEP x_s has the same set of type-1 equilibrium points for all $Q \in S^n_{++}$.*

Proof It is equivalent to say that the stability boundary of x_s has the same set of type-1 equilibrium points in both systems (2.3) and (2.8). Considering S^n_{++} is a convex open set, it is obvious that $\{Q_\lambda : Q_\lambda = \lambda I + (1-\lambda)Q, \lambda \in [0,1]\} \subset S^n_{++}$. Since for every $Q_\lambda \in S^n_{++}$, the assumptions A1) to A3) are satisfied for the quasi-gradient system (2.8), by Lemma 2.3.4, we know that the stability boundaries $\partial A_I(x_s)$ and $\partial A_Q(x_s)$ have the same set of type-1 equilibrium points. The theorem follows. ■

Among all type-1 equilibrium points on the stability boundary, we have special interest in the closest UEP and the controlling UEP. In the following theorems, we show that these two UEPs are also invariant in the quasi-gradient system as the matrix Q changes. We first show the invariance of the closest UEP.

Theorem 2.3.5 (Invariance of the closest UEP) *The quasi-gradient dynamical system (2.8) has the same closest UEP x_c with respect to the SEP x_s.*

Proof It is known from Lemma 2.3.1 that the objective function $f(x)$ is a common energy function for the system (2.8) for every $Q \in S^n_{++}$. By definition, the

closest UEP is the lowest energy equilibrium point on $\partial A_Q(x_s)$. Hence, we have $f(x_c) = \min_{x \in E \cap \partial A(x_s)} f(x)$. On the other hand, according to Theorem 2.3.2, $\partial A_Q(x_s)$ is the closure of the union of the stable manifolds of the boundary type-1 equilibrium points. In addition, the energy value $f(x)$ is decreasing along the trajectory. Hence, the closest UEP x_c must also be a type-1 equilibrium point. Moreover, x_c also has the lowest energy among all type-1 equilibrium points on $\partial A_Q(x_s)$.

On the other hand, according to Theorem 2.3.4, x_s has the same set of boundary type-1 equilibrium points in quasi-gradient systems for all Q. In other words, x_c will be the lowest energy type-1 equilibrium point on $\partial A_Q(x_s)$ for all Q. The theorem follows. ■

Now we show the invariance of the controlling UEP. Theorem 2.3.6 first shows that such invariance is hold locally.

Theorem 2.3.6 (Local invariance of the controlling UEP) *Suppose $x_p \in A_Q(x_s)$, $x_p \neq x_s$ and we can construct a search path $R(\lambda)$ such that $R(0) = x_p$ and $f(R(\lambda))$ is increasing. If $d(Q)$ satisfies assumptions A1) to A3), then there exists $\epsilon_Q > 0$ such that for every $\overline{Q}$ satisfying $|\overline{Q} - Q| \leq \epsilon_Q$, the controlling UEP w.r.t. the search path $R(\lambda)$ will be the same in $d(\overline{Q})$.*

Proof First we consider the trivial case where $x_e \in E \cap \partial A_Q(x_s)$, that is, the exit point is one of the boundary equilibrium points. According to Theorem 2.3.3, there exists $\epsilon_Q > 0$ such that for every $\overline{Q}$ satisfying $\|\overline{Q} - Q\| < \epsilon_Q$, every equilibrium point on $\partial A_Q(x_s)$ is also on $\partial A_{\overline{Q}}(x_s)$. In other words, $x_e \in \partial A_{\overline{Q}}(x_s)$. Meanwhile, x_e is on the search path $R(\lambda)$ by construction. Hence, x_e is the controlling UEP w.r.t. $R(\lambda)$ in $d(\overline{Q})$.

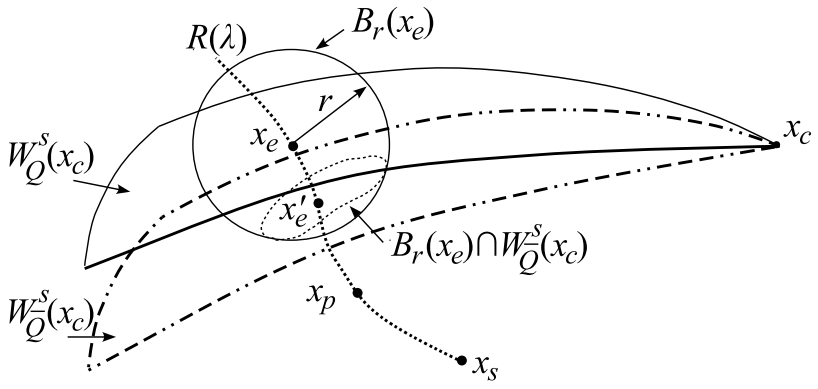


Figure 2.2: Illustration of the local invariance of the controlling UEP.

Now let $x_e \notin E \cap \partial A_Q(x_s)$ and $x_c \in E \cap \partial A_Q(x_s)$ be the controlling UEP w.r.t. to $R(\lambda)$ in $d(Q)$. Therefore, $x_e \in W^s_Q(x_c)$. Theorem 2.2.2 implies that there exist $r > 0$ such that $B_r(x_e) \bigcap \partial A_Q(x_s) = B_r(x_e) \bigcap W^s_Q(x_c)$. In fact, there exist $\tilde{r} > 0$ and $\epsilon_{\tilde{Q}} > 0$, such that for every $\tilde{Q}$ satisfying $\|\tilde{Q} - Q\| < \epsilon_{\tilde{Q}}$, $B_{\tilde{r}}(x_e) \bigcap \partial A_{\tilde{Q}}(x_s) = B_{\tilde{r}}(x_e) \bigcap W^s_{\tilde{Q}}(x_c)$. If this is not true, that is, for every $\tilde{r} > 0$ and $\epsilon_{\tilde{Q}} > 0$, there exist $\tilde{x}$ such that $\tilde{x} \in B_{\tilde{r}}(x_e) \bigcap \partial A_{\tilde{Q}}$ but $\tilde{x} \notin B_{\tilde{r}}(x_e) \bigcap W^s_{\tilde{Q}}(x_c)$. Let $\{r_1, r_2, \cdots\}$ and $\{\epsilon_1, \epsilon_2, \cdots\}$ be decreasing series satisfying $r_i \to 0$ and $\epsilon_i \to 0$ as $i \to \infty$. We can construct a corresponding series $\{x_1, x_2, \cdots\}$ satisfying $x_i \in B_{r_i}(x_e) \bigcap \partial A_{Q_i}$ but $x_i \notin B_{r_i}(x_e) \bigcap W^s_{Q_i}(x_c)$ for all x_i, where $\|Q_i - Q\| < \epsilon_i$. Obviously, $x_i \to x_e$ as $i \to \infty$, which implies that $x_e \notin W^s_Q(x_c)$. This leads to a contradiction.

On the other hand, Theorems 2.3.1 through 2.3.4 imply that there is no bifurcation in the quasi-gradient system (2.8) as Q changes. In other words, the topological structure of the system will be preserved as Q changes. Since $R(\lambda)$ intersects with $\partial A_Q(x_s)$ transversally at x_e and $R(\lambda)$ is continuous, there exist $\hat{r} > 0$ such that for every $0 < r < \hat{r}$ there are two points $x_1 = R(\lambda_1) \in A_Q(x_s)$ and $x_2 = R(\lambda_2) \notin A_Q(x_s)$, where x_1 and x_2 are the two intersection points between $R(\lambda)$ and $\partial B_r(x_e)$; meanwhile, $R(\lambda_3) \in B_r(x_e)$ for every $\lambda_3 \in (\lambda_1, \lambda_2)$. In other

words, x_1 is *inside* $A_Q(x_s)$ while x_2 is *outside* $A_Q(x_s)$. Considering that the stability boundary $\partial A_Q(x_s)$ changes continuously as Q changes, which is also a topology-preserving transform, there exists $\epsilon_{\hat{Q}}$ such that $x_1 \in A_{\hat{Q}}(x_s)$ and $x_2 \notin A_{\hat{Q}}(x_s)$, for every $\|\hat{Q} - Q\| < \epsilon_{\hat{Q}}$. Therefore, $R(\lambda)$ must intersect with $\partial A_{\hat{Q}}(x_s)$ in between of x_1 and x_2; in other words, there exists λ_3 satisfying $\lambda_1 < \lambda_3 < \lambda_2$, such that $x_3 = R(\lambda_3) \in \partial A_{\hat{Q}}(x_s)$. Apparently, $|x_3 - x_e| < \hat{r}$. Hence, x_3 is the exit point in the system $d(\hat{Q})$.

Now let $\overline{r} = \min\{\tilde{r}, \hat{r}\}$ and $\epsilon_Q = \min\{\epsilon_{\tilde{Q}}, \epsilon_{\hat{Q}}\}$. Therefore, for every $\|\overline{Q} - Q\| < \epsilon_Q$, there exists x'_e satisfying $x'_e \in R(\lambda) \bigcap \partial A_{\overline{Q}}(x_s)$ and $x'_e \in B_{\overline{r}}(x_e) \bigcap W^s_{\overline{Q}(x_c)}$. Hence, by definition, x_c is the controlling UEP w.r.t. $R(\lambda)$ in $d(\overline{Q})$. ■

The global invariance is shown in Theorem 2.3.7. To facilitate the proof of Theorem 2.3.7, Lemma 2.3.5 is given first to show the path-wise invariance of the controlling UEP.

Lemma 2.3.5 (Path-wise invariance of the controlling UEP) *Suppose $x_p \in A_Q(x_s)$ and we can construct a search path $R(\lambda)$ such that $R(0) = x_p$ and $f(R(\lambda))$ is increasing. Let $q(\lambda) = Q_\lambda : [0, 1] \to S^n_{++}$ be a continuous function, and $q(0) = Q_1$ and $q(1) = Q_2$. If $d(Q_\lambda)$ satisfies assumptions A1) to A3) for all $\lambda \in [0, 1]$, the controlling UEP w.r.t. the search path $R(\lambda)$ will be the same in systems $d(Q_1)$ and $d(Q_1)$.*

Proof Since $d(Q_\lambda)$ satisfies assumptions $(A1)$ and $(A2)$, $\forall \lambda \in [0, 1]$, it follows from Theorem 2.3.6 that there exist an $\epsilon_\lambda > 0$, such that x_p has the same controlling UEP in $d(\tilde{Q})$, for every $\tilde{Q}$ satisfying $\|\tilde{Q} - Q_\lambda\| < \epsilon_\lambda$. It is obvious that the set $\mathcal{B} = \{B_{\epsilon_\lambda}(Q_\lambda), \lambda \in [0, 1]\}$ forms an open cover of the set $\mathcal{Q} = \{Q_\lambda,\ \lambda \in [0, 1]\}$. On the other hand, since $\mathcal{Q}$ is a compact set, every open cover of $\mathcal{Q}$ has a finite subcover.

Therefore, there exist $\hat{\mathcal{B}} = \{B_{\epsilon_{\lambda_i}}(Q_{\lambda_i}), \lambda_i \in [0,1], i = 1, \cdots, k\}$ being a finite cover of $\mathcal{Q}$, such that $\mathcal{Q} \nsubseteq \hat{\mathcal{B}} - B_{\epsilon_{\lambda_i}}(Q_{\lambda_i})$, $\forall i = 1, \cdots, k$.

In addition, the elements in $\hat{\mathcal{B}}$ can be reordered such that $Q_1 \in B_{\epsilon_{\tilde{\lambda}_1}}$, $Q_2 \in B_{\epsilon_{\tilde{\lambda}_k}}$ and $B_{\epsilon_{\tilde{\lambda}_i}} \cap B_{\epsilon_{\tilde{\lambda}_{i+1}}} \neq \emptyset$, $\forall i = 1, \cdots, k-1$. Let $\hat{Q}_i \in B_{\epsilon_{\tilde{\lambda}_i}} \cap B_{\epsilon_{\tilde{\lambda}_{i+1}}}$, then the search path $R(\lambda)$ has the same controlling UEP in systems $d(Q_1)$, $d(\hat{Q}_1), \cdots, d(\hat{Q}_{k-1})$, $d(Q_2)$. Specifically, $R(\lambda)$ has the same controlling UEP in systems $d(Q_1)$ and $d(Q_2)$. ■

Theorem 2.3.7 (Global invariance of the controlling UEP) *Suppose $x_p \in A_Q(x_s)$ and we can construct a search path $R(\lambda)$ such that $R(0) = x_p$ and $f(R(\lambda))$ is increasing. If assumptions A1) to A3) are satisfied, the controlling UEP w.r.t. the search path $R(\lambda)$ will be the same in the quasi-gradient systems (2.8) for all $Q \in S^n_{++}$.*

Proof Considering that $\mathcal{S}^n_{++}$ is a convex open set, it is obvious that $\{Q_\lambda : Q_\lambda = \lambda I + (1-\lambda)Q,\ \lambda \in [0,1]\} \subset \mathcal{S}^n_{++}$. Since for every $Q_\lambda \in \mathcal{S}^n_{++}$, the assumptions (A1) to (A4) are satisfied for the pseudo-gradient system (2.8), by Lemma 2.3.5, we know that the controlling UEP w.r.t. the search path $R(\lambda)$ will be the same in $d(I)$ and $d(Q)$. The theorem follows. ■

2.4 Algorithms for checking invariant convergence

Based on the previous analyses of invariant properties in the quasi-gradient systems, we will study the invariance of partial stability region in the quasi-gradient system. Specifically, two types of invariance are developed in terms of the closest UEP and the controlling UEP, respectively. Each type of invariance will lead to a corresponding method for checking whether the trajectory start-

ing from a point will converge to the same stable equilibrium point in the quasi-gradient system, independent of the change of the matrix Q. Similar techniques have been widely used in direct methods for power system stability analysis [40, 41].

Before algorithms are proposed for checking invariant convergence of the trajectory in the quasi-gradient system, thus to answer the question raised in the beginning of the paper, we first introduce two theorems describing sufficient conditions for such invariant convergence. Theorem 2.4.1 is the sufficient condition considering the closest UEP as the reference, while Theorem 2.4.2 uses the controlling UEP

Theorem 2.4.1 (Invariant convergence w.r.t. the closest UEP) *Let* $x_c \in E \cap \partial A(x_s)$ *be the closest UEP with respect to* x_p. *If* $f(x_p) < f(x_c)$, *then* $x_p \in A_Q(x_s)$, *for every* $Q \in \mathcal{S}^n_{++}$.

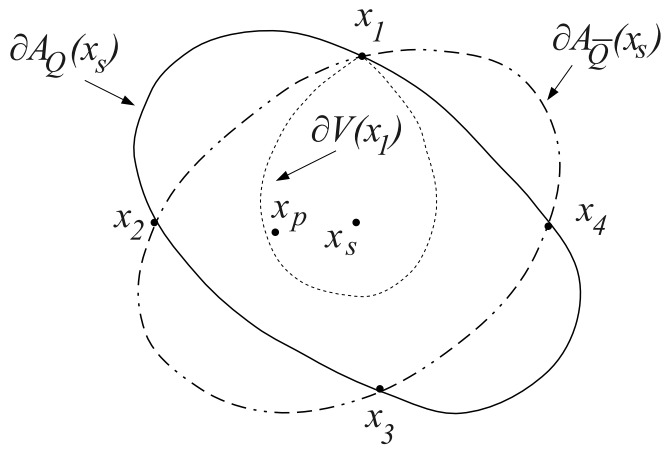


Figure 2.3: Illustration of the invariant convergence w.r.t. the closest UEP

Proof We already know that $f(x)$ is a common energy function for the quasi-gradient system (2.8). Let $v_c = f(x_c)$ and $C(v)$ denote the connected component

of the set $\{x : f(x) \leq v\}$ containing the s.e.p. x_s. We first show that $C(v_c) \subset A_Q(x_s)$, for every $Q \in \mathcal{S}^n_{++}$. It is known that v_c is actually the minimum of $f(x)$ over the stability boundary. Suppose there is a point $y \in C(v_c)$ with $f(y) < v_c$ and $y \notin \overline{A_Q(x_s)}$. Since $C(v_c)$ is a connected set, there must exist a point, say $\hat{y}$, such that $\hat{y} \in \partial A_Q(x_s)$ and $\hat{y} \in C(v_c)$. Therefore, $f(\hat{y}) < v_c$. But v_c is the minimum value of $f(x)$ on $\partial A_Q(x_s)$; hence we reach a contradiction. This implies $R(v_c) \subset A_Q(x_s)$.

Now since $f(x_p) < f(x_c)$ and $x_p \in A(x_s)$, hence $x_p \in C(v_c)$. In other words, $x_p \in A_Q(x_s)$, for every $Q \in \mathcal{S}^n_{++}$. The theorem follows. ■

Theorem 2.4.2 (Invariant convergence w.r.t. the controlling UEP) *Suppose $x_p \in A(x_s)$ and we can construct a search path $R(\lambda)$ such that $R(0) = x_p$ and $f(R(\lambda))$ is increasing. Let $x_c \in E \cap \partial A(x_s)$ be the controlling UEP with respect to $R(\lambda)$. If $f(x_p) < f(x_c)$, then $x_p \in A_Q(x_s)$, for every $Q \in S^n_{++}$.*

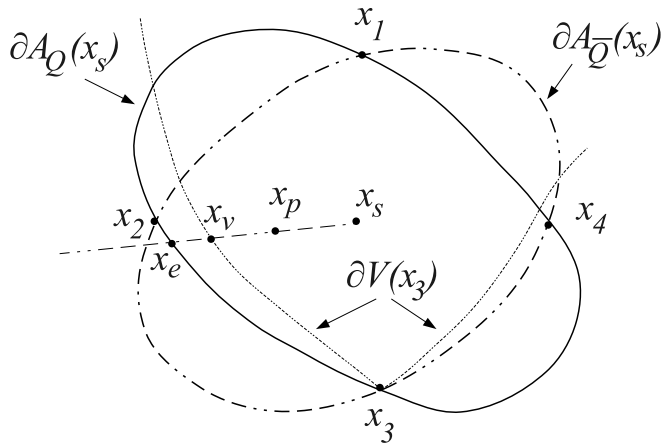


Figure 2.4: Illustration of the invariant convergence w.r.t. the controlling UEP

Proof Let $v_c = f(x_c)$ and $C(v)$ denote the connected component of the set $\{x : f(x) \leq v\}$ containing the SEP x_s. Let x_λ be the intersection of the search path $R(\lambda)$

and $\partial C(x_c)$, and let x_e^Q be the exit point w.r.t. x_p in the quasi-gradient system $d(Q)$, as illustrated in Fig. 2.4.

According to Theorem 2.3.7, x_c is also the controlling UEP with respect to the search path $R(\lambda)$ in $d(Q)$. Hence, the exit point $x_e^Q \in W_Q^s(x_c)$, and we have $f(x_c) < f(x_e^Q)$. Since $f(x_v) = f(x_c)$, we have $f(x_v) < f(x_e^Q)$. Because $f(R(\lambda))$ is increasing w.r.t. λ on the search path $R(\lambda)$, we know that the search path $R(\lambda)$ will meet $\partial C(x_c)$ before meeting $\partial A_Q(x_s)$. Hence, if $f(x_p) < f(x_c)$, that is $f(x_p) < f(x_v)$, then it is ensured that x_p is within the stability boundary $\partial A_Q(x_s)$; in other words, $x_p \in A_Q(x_s)$. The theorem follows. ■

Now we are at the place to introduce the algorithms for checking whether the trajectory starting from an initial point in the quasi-gradient system will converge to the same SEP. Similarly, two versions of algorithms are proposed, based on the closest and the controlling UEP, respectively.

Algorithm 1: checking invariant convergence based on the closest UEP

Input: An initial point x_0.

Output: Judgement regarding the invariant convergence of the trajectory starting from x_0 in the quasi-gradient system (2.8).

Algorithm:

Step 1.1 Integrate the gradient system (2.3) starting from x_0 and compute the original stable equilibrium point x_s.

Step 1.2 Compute all type-1 equilibrium points on the stability boundary $\partial A(x_s)$,

denoted as x_i, $i = 1, , k$.

Step 1.3 Determine the closest UEP x_c with respect to x_s, which is

$$x_c = \arg\min_x \{ f(x) : x \in \{x_1, \cdots, x_k\}\}. \tag{2.32}$$

Step 1.4 Compare the objective value of x_0 with that of x_c. If $f(x_0) < f(x_c)$, we can output the judgement that the trajectory starting from x_0 will converge to x_s in the quasi-gradient system (2.8), which is independent of the change of the matrix Q.

In practice, however, it is very difficult or even impossible to compute all type-1 equilibrium points on the stability boundary $\partial A(x_s)$. In addition, because the closest UEP is the lowest energy equilibrium point on $\partial A(x_s)$, the resulting judgement in Step 1.4 could be very conservative. Instead of using the closest UEP, Algorithm 2 uses the energy value of the controlling UEP as the criterion to reduce conservativeness. In the meantime, effective numerical methods are available for computing the controlling UEP, which make the algorithm appealing for practical applications.

Algorithm 2: checking invariant convergence based on the controlling UEP

Input: An initial point x_0, and a search path $R(\lambda)$ with $R(0) = x_0$.

Output: Judgement regarding the invariant convergence of the trajectory starting from x_0 in the quasi-gradient system (2.8).

Algorithm:

Step 2.1 Integrate the gradient system (2.3) starting from x_0 and compute the original stable equilibrium point x_s.

Step 2.2 Compute the controlling UEP x_c with respect to the search path $R(\lambda)$. The BCU method proposed in [41] or the method reported in [102] can be used for computing the controlling UEP.

Step 2.3 Compare the objective value of x_0 with that of x_c. If $f(x_0) < f(x_c)$, we can output the judgement that the trajectory starting from x_0 will converge to x_s in the quasi-gradient system (2.8), which is independent of the change of the matrix Q.

It needs to be noted that, since the conditions described in Theorem 2.4.1 and Theorem 2.4.2 are sufficient, the judgement obtained by the above two algorithms are conservative. In other words, if the above conditions are violated, no conclusion can be made regarding convergence of the trajectory in the quasi-gradient system as the matrix Q changes. Starting from an arbitrary initial point, the following algorithm is proposed to ensure convergence to the same solution, regardless of the numerical operation imposed on the dynamical system.

Algorithm 3: ensuring invariant convergence from an initial point

Input: An arbitrary initial point x_0, and a search path $R(\lambda)$ with $R(0) = x_0$.

Target: The trajectory starting from x_0 will converge to the same solution regardless of Q in the quasi-gradient system (2.8).

Algorithm:

Step 3.1 Compute the closest or controlling UEP x_c in the original dynamical system (2.3).

Step 3.2 Compare the objective value of x_0 with that of x_c. If $f(x_0) < f(x_c)$, the trajectory starting from x_0 will converge invariantly to the same solution, go to Step 3.4; otherwise, go to Step 3.3.

Step 3.3 Integrate the gradient system (2.3) starting from x_0 until a point x_1 is reached such that $f(x_1) < f(x_c)$. In other words, the trajectory starting from x_1 will converge invariantly to the same solution.

Step 3.4 Apply the appropriate numerical operation that can speed-up movement of the trajectory and fast compute the solution x_s in the quasi-gradient system (2.8).

2.5 Numerical Simulation

In this section, numerical simulations are carried out to validate the proposed algorithms. Consider the following minimization problem:

$$\min_x \; f(x) \;=\; -2\cos(x_1) - \cos(x_2) - \cos(x_1 - x_2) - 0.1x_1 - 0.3x_2 + 0.4. \tag{2.33}$$

To solve this problem using TRUST-TECH, the following negative gradient system is first constructed:

$$\begin{aligned} \dot{x}_1 &= -2\sin(x_1) - \sin(x_1 - x_2) + 0.1 \\ \dot{x}_2 &= -\sin(x_2) - \sin(x_2 - x_1) + 0.3 \end{aligned}. \tag{2.34}$$

For illustration, all equilibrium points located within the range $-10 \leq x_1, x_2 \leq 10$ are computed by solving the following system of nonlinear equations with different starting points:

$$\begin{aligned} -2\sin(x_1) - \sin(x_1 - x_2) + 0.1 &= 0 \\ -\sin(x_2) - \sin(x_2 - x_1) + 0.3 &= 0 \end{aligned} . \tag{2.35}$$

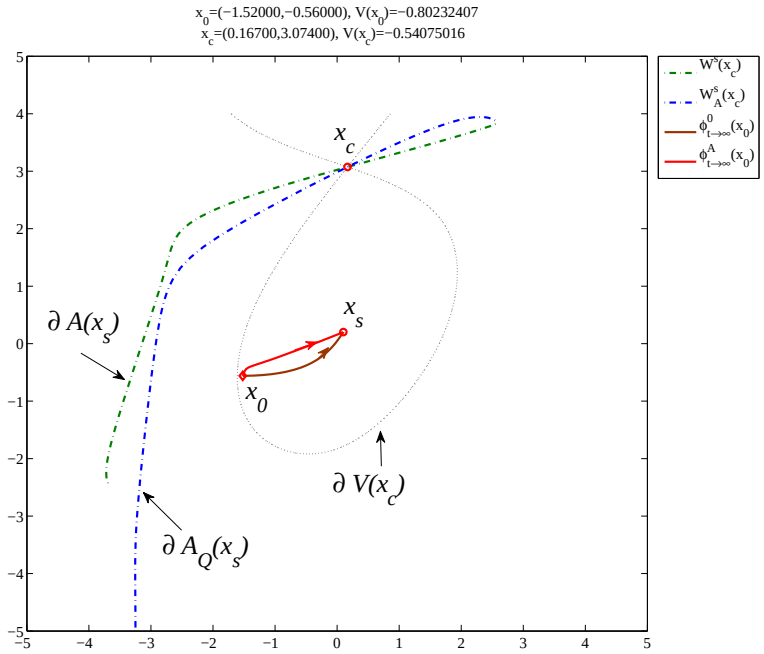
The quasi-gradient system is defined by multiplying randomly generated positive-definite matrices Q to the system (2.34). Convergence of trajectories from different points in the stability region of the SEP $x_s = (0.10, 0.20)$ in the original gradient system (2.34) are studied as the matrix Q is changed. The type-1 UEP $x_c = (0.167, 3.074)$ has the lowest objective value on the stability boundary of x_s, thus it is the closest UEP with $f(x_c) = -0.54$.

2.5.1 Convergence to the same SEP

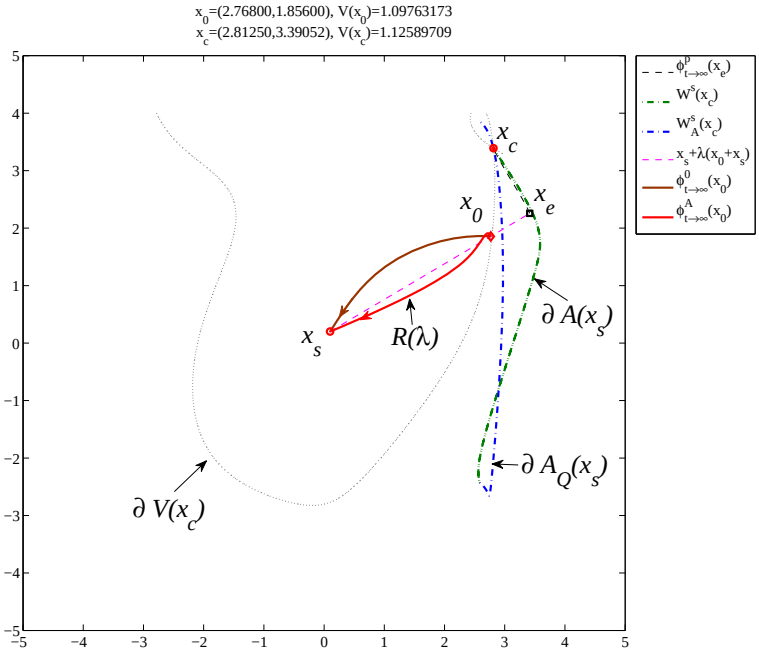
We first show that when the proposed criteria are satisfied, the trajectory from an initial point will still converge to x_s in the quasi-gradient system.

The invariant convergence in terms of the closest UEP is first illustrated. The initial point is $x_0 = (-1.52, -0.56)$, its objective value is $f(x_0) = -0.802$. This objective (energy) value is lower than that of the closest UEP. Hence, by Theorem 2.4.1, the point x_0 is within the invariant part of the stability region and the solution trajectory starting from x_0 will always converge to x_s no mater how the matrix Q is changed. Fig. 2.5(a) illustrates such invariant convergence when the multiplied matrix is:

$$Q = \begin{bmatrix} 0.173 & 0.606 \\ 0.606 & 10.537 \end{bmatrix}. \tag{2.36}$$



(a) In terms of the closest UEP



(b) In terms of the constrolling UEP

Figure 2.5: Illustration of the invariant convergence

The eigenvalues of this matrix are $\lambda_1 = 0.138$ and $\lambda_2 = 10.572$. In Fig. 2.5(a), the solid curves are the contour plot of the objective function $f(x)$, the red and blue dashed curves passing the closest UEP x_c are segments of stability boundary in the original gradient system (2.34) and the quasi-gradient system after multiplying with Q, respectively. The bold and dashed green and red curves are the trajectories starting from x_0 in the two systems, respectively. Obviously, the two trajectories both converge to the same SEP x_s.

The invariant convergence in terms of the controlling UEP is illustrated with the help of Fig. 2.5(b). The initial point is $x_0 = (2.77, 1.86)$, its objective value is $f(x_0) = 1.098$. The search path is constructed as the line emanated from x_s and passing x_0, that is

$$R(\lambda) = x_s + (1 + \lambda)(x_0 - x_s). \tag{2.37}$$

The exit point is $x_e = (3.42, 2.26)$, which is approximated as the first local maximum of $f(x)$ along $R(\lambda)$. Using the method reported in [102], the corresponding controlling UEP is computed, which is $x_c = (2.81, 3.39)$ with $f(x_c) = 1.126$. Since $f(x_0) < f(x_c)$, by Theorem 2.4.2, the point x_0 is within the invariant part of the stability region and the solution trajectory starting from x_0 will always converge to x_s no mater how the matrix Q is changed. Fig. 2.5(b) illustrates such invariant convergence when the multiplied matrix is the same as that in Fig. 2.5(a):

$$Q = \begin{bmatrix} 0.173 & 0.606 \\ 0.606 & 10.537 \end{bmatrix}. \tag{2.38}$$

The trajectories starting from x_0 converge to the same SEP x_s in both the original gradient system and the quasi-gradient system.

2.5.2 Convergence to different SEPs

Now we show that, if the proposed conditions are violated, trajectories in the quasi-gradient system could converge to different stable equilibrium points other than the original one x_s.

Violation of the invariant convergence is first shown in terms of the closest UEP. The initial point is $x_0 = (-3.50, -1.50)$, its objective value is $f(x_0) = 3.418$. This objective (energy) value is greater than that of the closest UEP $f(x_c) = -0.541$. As illustrated in Fig. 2.6(a), the trajectory starting from x_0 converges to x_s in the original gradient system. However, in the quasi-gradient system with the same multiplying matrix

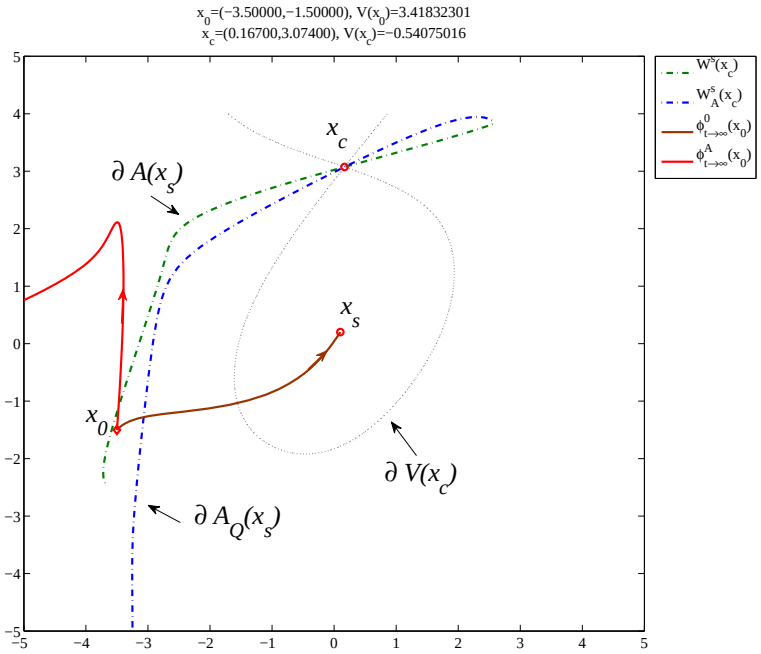
$$Q = \begin{bmatrix} 0.173 & 0.606 \\ 0.606 & 10.537 \end{bmatrix}, \tag{2.39}$$

the point x_0 is no more located in the stability region of x_s. Hence, the trajectory starting from x_0 now converges to another stable equilibrium point $x_s' = (-6.18, -0.20)$, as shown in Fig. 2.6(a).

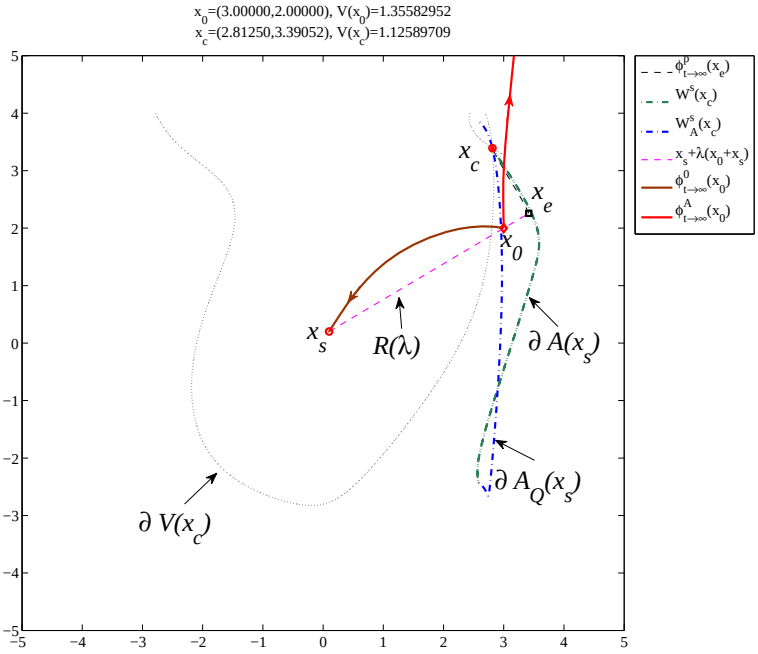
Fig. 2.6(b) illustrates violation of the invariant convergence in terms of the controlling UEP. The initial point is $x_0 = (3.00, 2.00)$, its objective value is $f(x_0) = 1.36$. The search path is constructed as the line emanated from x_s and passing x_0, that is

$$R(\lambda) = x_s + (1 + \lambda)(x_0 - x_s). \tag{2.40}$$

The exit point and the corresponding controlling UEP are the same as that are shown in Fig. 2.5(b), that is, $x_e = (3.42, 2.26)$ and $x_c = (2.81, 3.39)$. It is obvious that $f(x_0) > f(x_c) = 1.12$. As shown in Fig. 2.6(b), in the quasi-gradient system, the trajectory starting from x_0 now converges to another stable equilibrium



(a) In terms of the closest UEP



(b) In terms of the constrolling UEP

Figure 2.6: Illustration of the convergence to different SEPs

point $x'_s = (6.38, 12.77)$.

2.6 Summary

In this chapter, we study the relationship between a gradient dynamical system $\dot{x} = -\nabla f(x)$ and its associated quasi-gradient system $\dot{x} = -Q\nabla f(x)$ with $Q \in S^n_{++}$. We show that, under a few generic assumptions, partial stability region in the quasi-gradient system is invariant of the change of Q. Such invariance has been characterized in terms of the closest and the controlling UEPs. Corresponding to these invariant properties, sufficient conditions and methods are proposed for checking and preserving the invariant convergence of the trajectory starting from a given point in the quasi-gradient system. It needs to be emphasized that applicability of the theoretical results and algorithms developed in this paper is not confined to TRUST-TECH based methods or general trajectory based methods. These results and methods can also be applied as analytical tools for convergence analysis of other numerical methods.

CHAPTER 3

NONLINEAR FEASIBILITY ANALYSIS USING TRUST-TECH

3.1 Introduction

A standard optimization model consists of two primary parts, an objective function and a set of linear or/and nonlinear constraint functions, as shown in (3.1):

$$\begin{aligned} \min \quad & f(x) \\ s.t. \quad & h_i(x) = 0,\ i \in \mathcal{E} = \{1, \cdots, n_{\mathcal{E}}\} \\ & g_j(x) \le 0,\ j \in \mathcal{I} = \{1, \cdots, n_{\mathcal{I}}\}, \end{aligned} \tag{3.1}$$

where, $\mathcal{E}$ is the index set for equality constraints $h_i(x)$ and $\mathcal{I}$ is that for inequality constraints $g_j(x)$. The target of the optimization problem (3.1) is to find values for the variables x that provide a minimum value for $f(x)$, while all restrictions represented by the constraint functions $h(x)$ and $g(x)$ can also be satisfied. Compared with unconstrained optimization problems, the existence of nonlinear constraints in the optimization problem (3.1) makes the task much more complicated. Analysing feasibility of the problem and attaining a feasible solution is usually the first step to be carried out in solving the constrained nonlinear program (3.1) [18, 43].

Indeed, the task of feasibility computation, that is, computing a solution point x to satisfy the set of nonlinear equality and inequality constraints

$$\begin{aligned} & h_i(x) = 0,\ i \in \mathcal{E} = \{1, \cdots, n_{\mathcal{E}}\} \\ & g_j(x) \le 0,\ j \in \mathcal{I} = \{1, \cdots, n_{\mathcal{I}}\} \end{aligned} \tag{3.2}$$

is itself a challenging problem. This class of constraint satisfaction problems appears in a wide variety of problems in engineering, the sciences and applied

mathematics. For example, power flow analysis [15, 150, 163], one of the fundamental tasks in power system computation, falls into this category of problems.

More often than not, however, there does not always exist a solution to the problem (3.2). In other words, the set of feasible points satisfying all the constraints (3.2) could be empty. On the other hand, different constraints can have different levels of priority. As a compromise, those low-priority constraints (soft constraints) can be relaxed such that a feasible solution can still be obtained. Identifying the most sensitive constraints and restoring feasibility by adjusting these constraints are of special interest in practical applications.

In this chapter, TRUST-TECH based methods are developed for solving the feasibility analysis problem (3.2). Following the spirit in the TRUST-TECH methodology, the problem is solved with the aid of a particular nonlinear dynamical system. Specifically, finding feasible points to satisfy the set of nonlinear constraints is accomplished via finding all stable equilibrium manifolds in the constructed dynamical system. For infeasible problems, a TRUST-TECH based method is developed to restore feasibility with a two-phase strategy. It first finds the global minimum-violation point via systematically finding all local minimum-violation points; then it restores feasibility by optimally adjusting constraints at the attained global minimum-violation point.

The proposed methods are applied to solve practical problems in power system analysis. As an efficient numerical implementation, the pseudo-transient continuation method is employed in the proposed methods to achieve a fast convergence. The first problem to be solved is power flow computation. The key features of TRUST-TECH based power flow computation are twofold: 1) if the system possesses a power flow solution, the method can converge globally

to the power flow solution; and 2) if the system does not possess a power flow solution, instead of divergence, the method will converge to a solution point with useful information for guiding feasibility restoration. Numerical simulations have been carried out on power systems of 30 and 2383 buses. The second problem is the optimal power flow (OPF) problem, where the TRUST-TECH based method is used to restore feasibility for OPF problems with infeasible constraints. Numerical simulations are carried on a 300-bus system with promising results. These practical results have shown effectiveness of the proposed TRUST-TECH based feasibility analysis methods.

3.2 Feasibility Analysis

Before we propose the TRUST-TECH based methods for feasibility analysis, a brief review of the target problems is given in this section.

There are several good reasons for the need to reach feasibility quickly and reliably in solving the nonlinear optimization problem (3.1). Firstly, some solution methods are unable to proceed to optimality without first reaching a feasible solution, which is not uncommon for nonlinear optimization methods [20, 56]. Secondly, overall solution speed can be increased in some algorithms if a feasible solution is available, as happens in branch and bound methods for mixed-integer programs [72, 74]. Thirdly, a feasible solution is all that is required, for example, in power flow computations [48, 126, 163]. Finally, many methods for infeasibility analysis can be accelerated if the feasibility status of subsets of constraints can be decided quickly [43].

Finding a feasible point for a nonlinear program, however, can be a very

challenging task. There may be multiple disconnected feasible regions which could be located at extreme distances from each other. A very common approach to finding a feasible solution for the set of nonlinear constraints (3.2) is to minimize an unconstrained function that assigns a non-negative penalty for each constraint violation at a given point. In general, the penalty function reaches zero at a feasible point [53]. Minimizing such penalty function itself, however, can be a challenging global optimization problem. Therefore, many algorithms for finding a feasible point depend on special characteristics of the problem, such as convexity of the feasible region [43]. For problems whose characteristics can not be easily sought, heuristic techniques like multi-start and random sampling are also used to approximate the feasibility [42, 83]. Global optimization methods, especially evolutionary algorithms [116], have also been used for feasibility seeking with interesting results.

As optimization models grow larger and more complex, infeasibility happens more often during the process of model formulation,and is harder to diagnose than ever before. For nonlinear optimization problems, the issue becomes even more complicated. Specifically, the model might be feasible but the solver, especially a local one, is given a poor starting point from which it is unable to reach feasibility. For a truly infeasible model, a natural question raised is: which (single or multiple) constraints are causing the infeasibility and how would the feasibility be restored? Three main approaches have been developed in recent years for diagnosing and repairing infeasibility. The first one is developed to identify an *irreducible infeasible subset (IIS)* of constraints, which is usually implemented via sequential or floating constraint selection methods [43]. In contrast, the second method analyses infeasibility by identifying a *maximum feasible subset (MFS)* of constraints [2, 122]. Identifying an MFS, however, is an NP-hard

problem and success of these methods relies heavily on good heuristics. The last approach seeks to find a best way to alter constraints (e. g. the fewest changes to the constraints) for restoring feasibility. This approach can be implemented as shifting the right hand side of the constraints [21, 107, 180] or adjusting all of the constraint coefficients [3]. Most available methods, however, address only infeasible linear systems and very limited success has been achieved for nonlinear problems [43].

3.3 TRUST-TECH Based Methods

We assume that nonlinear equality constraints $h_i(x)$, $i \in \mathcal{E}$ and nonlinear inequality constraints $g_j(x)$, $j \in \mathcal{I}$ are all twice differentiable, that is, they all belong to $C^2 : R^n \rightarrow R$.

We first notice that inequalities in (3.2) can be replaced by equalities by adding slack variables as follows:

$$g_j(x) \leq 0 \quad \Rightarrow \quad g_j(x) + s_j^2 = 0, \; s_j \in R. \tag{3.3}$$

Therefore, without loss of generality, we consider the following feasibility problem with only equality constraints:

$$h_i(x) = 0, \; i = \{1, \cdots, m\}. \tag{3.4}$$

We define the following equality vector $H(x) := (h_1(x), \cdots, h_m(x))^T$. By applying the regularity condition and Sard's theorem [149], it can be shown that the constraint set (or the feasible set or region) defined by

$$M = \{x \in R^n : H(x) := (h_1(x), \cdots, h_m(x))^T = 0\} \tag{3.5}$$

is a smooth manifold. In general, the extremely complicated constraint set M can be decomposed into several disjoint path-connected feasible components, say

$$M = \bigcup_k M_k, \tag{3.6}$$

where, $M_i \cap M_j = \emptyset$, $\forall i \neq j$.

Assumption 3.3.1 *The constraint set $H \in C^2 : R^n \rightarrow R^m$ satisfies one of the following conditions:*

1. *$\|H(x)\|$ is a proper map (i.e., the pre-image of a compact set is compact), or*
2. *For any $\gamma > 0$ and for any closed subset K of*

$$\{x \in R^n : \|H(x)\| \leq \gamma,\ \|\nabla^T H(x) \cdot H(x)\| \neq 0\}, \tag{3.7}$$

we have

$$\inf \left\{ \nabla^T H(x) \cdot H(x) \| : x \in K \right\} > 0. \tag{3.8}$$

3.3.1 Feasibility computation

The TRUST-TECH based methodology has been developed to compute solutions to the feasibility computation problem (3.4) [103]. This methodology finds all the feasible components via exploring certain trajectories of a particular dynamical system. In order to visit each feasible component, the proposed method consists of two steps:

Step 1.1: Approach a (path-connected) feasible component of the feasibility computation problem (3.4).

Step 1.2: Move away from the feasible component and approach another feasible component of the feasibility computation problem (3.4).

We design a nonlinear dynamical system whose trajectories can be used to perform step 1.1 and 1.2. The central idea in designing such a nonlinear dynamical system is that every path-connected feasible component corresponds with a stable equilibrium manifold (a generalized concept of stable equilibrium point) of the nonlinear dynamical system. In this way, the task of finding all feasible components of the feasibility computation problem (3.4) is equivalent to the task of finding all stable equilibrium manifolds of the nonlinear dynamical system. To this end, we build the following so called *quotient gradient system (QGS)* whose vector field is associated with the constraint set M characterized by the equality vector $H(x)$:

$$\dot{x} \;=\; -\nabla^T H(x) \cdot H(x), \tag{3.9}$$

where,

$$\nabla H(x) = \begin{bmatrix} \frac{\partial h_1(x)}{\partial x_1} & \frac{\partial h_1(x)}{\partial x_2} & \cdots & \frac{\partial h_1(x)}{\partial x_n} \\ \frac{\partial h_2(x)}{\partial x_1} & \frac{\partial h_2(x)}{\partial x_2} & \cdots & \frac{\partial h_2(x)}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial h_m(x)}{\partial x_1} & \frac{\partial h_m(x)}{\partial x_2} & \cdots & \frac{\partial h_m(x)}{\partial x_n} \end{bmatrix} \tag{3.10}$$

represents the Jacobian matrix of the vector $H(x)$. An energy function associated with the QGS (3.9) can be defined as

$$V(x) \;=\; \frac{1}{2}\|H(x)\|^2. \tag{3.11}$$

Theorem 3.3.1 (Completely Stable [101, 104]) *If the feasibility computation problem (3.4) satisfies Assumption 3.3.1 and all the equilibrium manifolds are pseudo-hyperbolic and finite in number, then every trajectory of the QGS (3.9) is bounded and converges to one of the equilibrium manifolds.*

Theorem 3.3.2 (Equilibrium Manifolds and Feasible Components [101, 104]) *If the feasibility computation problem (3.4) satisfies Assumption 3.3.1 and all the equilibrium manifolds are pseudo-hyperbolic and finite in number, then each path-connected component of the constraint set M of the feasibility computation problem (3.4) is a stable equilibrium manifold of the QGS (3.9). Conversely, if Σ is a stable equilibrium manifold of the QGS (3.9), then Σ is a non-isolated set of local minima of the following minimization problem*

$$\min_{x \in R^n} \quad \frac{1}{2}\|H(x)\|^2. \tag{3.12}$$

Theorem 3.3.2 provides a basis for our proposed method in searching for feasible components of the feasibility computation problem (3.4). Hence, steps 1.1 and 1.2 can be numerically implemented via the following tasks:

Task 1.1: Approach a stable equilibrium manifold of the QGS (3.9).

Task 1.2: Escape from the stable equilibrium manifold and approach another stable equilibrium manifold of the QGS (3.9).

Task 1.1 can be numerically implemented by following the trajectory of the QGS (3.9) starting from any initial point, which is an infeasible point of the feasibility computation problem (3.4) to a point located in the stable manifold of system (3.9) (i.e. a point located in a feasible component of the feasibility computation problem (3.4)). Task 1.2 can be numerically implemented by following the trajectory in reverse time starting from an initial point in the stable manifold of the QGS (3.9), until the trajectory approaches a point in an unstable equilibrium manifold on the stability boundary of the stable equilibrium manifold. Then by following the trajectory starting from an initial point, which is close to

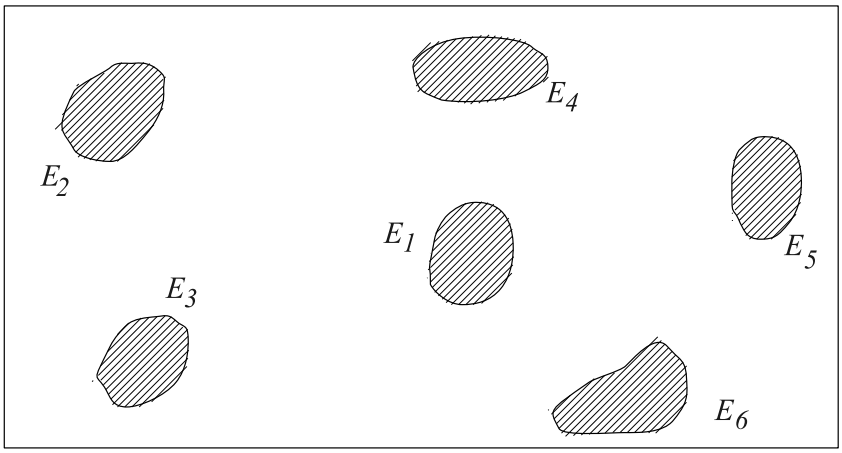


Figure 3.1: $E_1, E_2, \cdots, E_6$: stable equilibrium manifolds.

the unstable equilibrium manifold but outside the stability region, we can approach another stable equilibrium manifold of the QGS (3.9), which is another feasible component of the feasibility computation problem (3.4).

It needs to be noticed that, the correspondence between feasible components of the feasibility computation problem (3.4) and the stable equilibrium manifolds of the QGS (3.9) may not a bijective one. In particular, the one-to-one correspondence can only be defined between feasible components and stable equilibrium manifolds with zero energy, that is, $V(x) = 0$ for all points lying in the stable equilibrium manifolds.

We illustrate in Fig. 3.1 to Fig. 3.5 the search process of the TRUST-TECH based methodology. Suppose there are six feasible components in the search space of the feasibility computation problem (3.4) (Fig. 3.1). We define a QGS system described by the equation (3.9) which is completely stable and each stable equilibrium manifold corresponds with each feasible component of the feasibility computation problem (3.4). The dashed line represents the stability boundary of each stable equilibrium manifold (i.e., each feasible component)

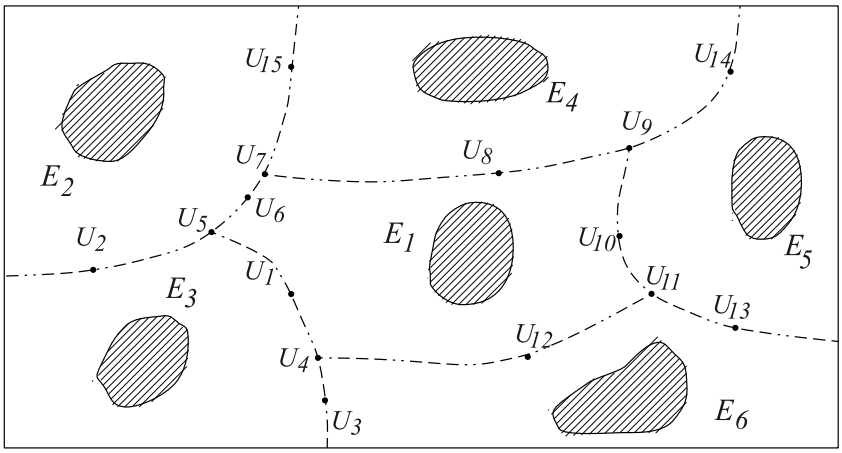


Figure 3.2: $E_1, E_2, \cdots, E_6$: stable equilibrium manifolds; $- \cdot -$: stability boundary (i.e. boundary of stability region); $U_1, \cdots, U_{15}$: unstable equilibrium manifolds.

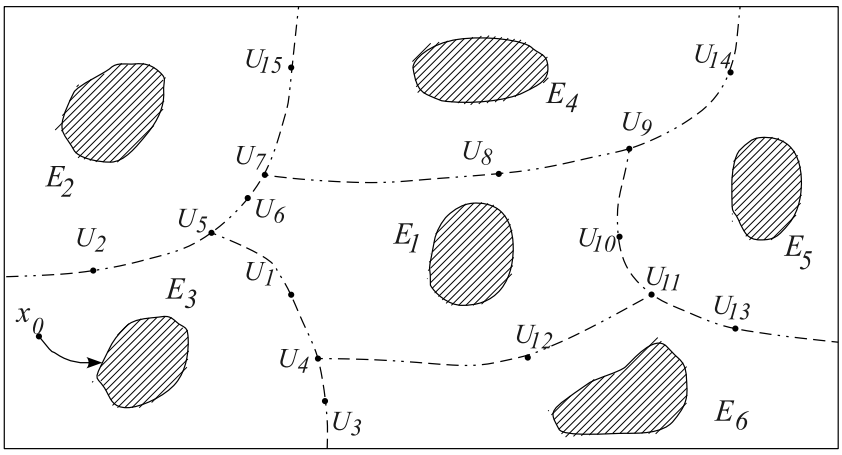


Figure 3.3: $E_1, E_2, \cdots, E_6$: stable equilibrium manifolds; $- \cdot -$: stability boundary (i.e. boundary of stability region); $U_1, \cdots, U_{15}$: unstable equilibrium manifolds; x_0: initial condition; $\rightarrow$: system trajectory.

(Fig. 3.2). Starting from an initial point, the system trajectory of the dynamical system (or a local search method) finds a stable equilibrium manifold (i.e., a feasible component) (Fig. 3.3). Starting from the found feasible component, the numerical method for computing tier-one feasible components computes three initial points that lie inside the stability region of each stable equilibrium

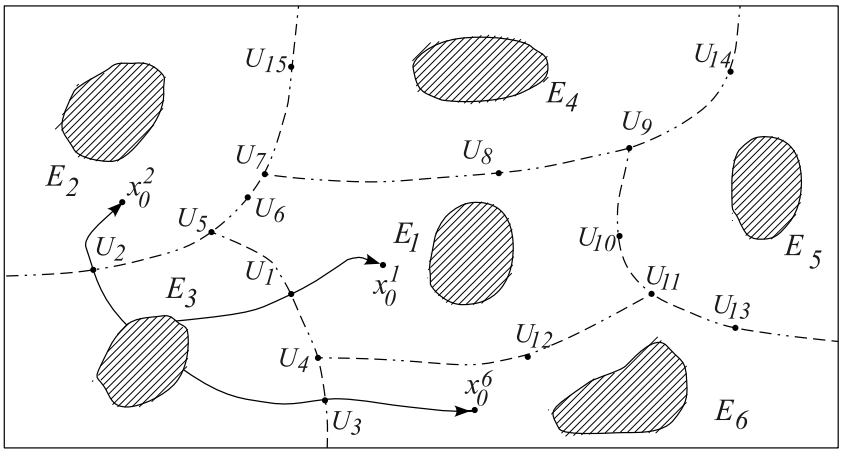


Figure 3.4: $E_1, E_2, \cdots, E_6$: stable equilibrium manifolds; $- \cdot -$: stability boundary (i.e. boundary of stability region); $U_1, \cdots, U_{15}$: unstable equilibrium manifolds; $\rightarrow$: search paths by TRUST-TECH.

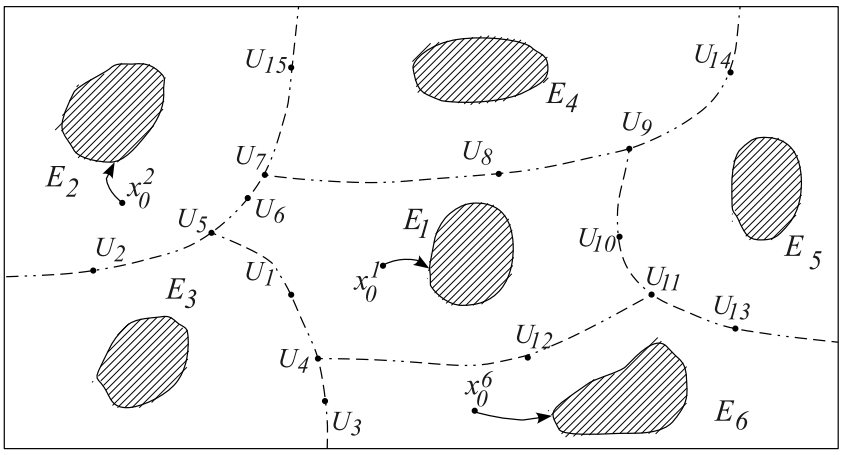


Figure 3.5: $E_1, E_2, \cdots, E_6$: stable equilibrium manifolds; $- \cdot -$: stability boundary (i.e. boundary of stability region); $U_1, \cdots, U_{15}$: unstable equilibrium manifolds; $\rightarrow$: system trajectory.

manifold. In addition, each initial point is close to the corresponding stable equilibrium manifold (i.e., a tier-one feasible component) (Fig. 3.4). Starting from an initial point, the corresponding (dynamical) trajectory or a local search method computes the corresponding stable equilibrium manifold (i.e., a feasible component) (Fig. 3.5). These figures serve to emphasize the capability of the TRUST-TECH based method in computing surrounding feasible components in

a tier-by-tier manner.

It has a good possibility that some equilibrium manifold of the QGS (3.9) does not correspond with a feasible component of the problem (3.4). In addition, a poor initial point can be provided for the feasibility computation. For example, assume now the stable equilibrium manifold E_3 is found to have non-zero energy value; hence, it is not a feasible component of the problem (3.4). As a result, local methods starting from the given initial point x_0, which lies in the stability region of E_3, will converge to an infeasible point lying in E_3 and not be able to escape from the attained point. Other existing heuristic methods, such as multi-start and random sampling techniques, can still fail to find a feasible component if the stability region of E_3 is large enough and the selected sampling points all happen to lie in it. As a consequence, these methods will report infeasibility of the problem (3.4), although it is truly feasible. The proposed TRUST-TECH based method eliminates such possibility of false negative errors by systematically locating all the stable equilibrium manifolds of the QGS (3.9), thus all feasible components of the feasibility computation problem (3.4).

3.3.2 Feasibility restoration

This TRUST-TECH based framework can also be used to analyse an infeasible problem (3.4), that is, there is no solution available to satisfy all the imposed constraints simultaneously. However, feasibility can still be achieved by adjusting low priority constraints of the infeasible problem. One reliable way to restore feasibility for an infeasible problem (3.4) is to first find the global minimum-violation component and then use it to guide the adjustment of the constraints

to achieve feasibility. To this end, we propose a TRUST-TECH based method for infeasibility analysis, which is composed of two distinct phases: In phase I, starting from an arbitrary starting point, it finds the global minimum-violation component by systematically finding all local minimum-violation components. In phase II, it restores the feasibility by optimally adjusting the constraints at the attained global minimum-violation component.

Phase I

The proposed TRUST-TECH based method for infeasibility analysis first finds the non-isolated set of global minima of the following violation minimization problem

$$\min_{x \in R^n} \quad \frac{1}{2}\|H(x)\|^2 \tag{3.13}$$

by finding all non-isolated sets of local minima in the search space with local minimum violation using some trajectories of a particular class of dynamical systems. It is obvious that the global minimum of the optimization problem (3.13) attains a positive objective for an infeasible problem (3.4); otherwise, the problem (3.4) is in fact feasible. This method is conceptually composed of two steps:

Step 2.1: Approach a local minimum violation component of the infeasible problem (3.4).

Step 2.2: Move away from the local minimum violation component and approach another local minimum violation component of the infeasible problem (3.4).

We design a nonlinear dynamical system whose trajectories can be used to perform step 2.1 and 2.2. The central idea in designing such a nonlinear dynamical system is that every path-connected set of local minimum violations corresponds with a stable equilibrium manifold of the nonlinear dynamical system. In this way, the task of finding all sets of local minimum-violation of the infeasibility problem (3.4) is equivalent to the task of finding all the stable equilibrium manifolds of the nonlinear dynamical system. Apparently, the same QGS (3.9) can serve as the desired dynamical system.

Theorem 3.3.2 also shows that if a set is a local minimum-violation component of the infeasible problem (3.4), then the set is a stable equilibrium manifold of the QGS (3.9). This analytical result provides a basis for our proposed method in searching for sets of local minimum-violation of the infeasible problem (3.4). Hence, steps 2.1 and 2.2 can be numerically implemented via the following tasks:

Task 2.1: Approach a stable equilibrium manifold of the QGS (3.9).

Task 2.2: Escape from the stable equilibrium manifold and approach another stable equilibrium manifold of the QGS (3.9).

Phase II

Phase II is designed to regain feasibility for the infeasible problem (3.4). Indeed, the global minimum-violation component found in phase I provides information regarding the minimum distance to feasibility. However, it provides no clue on how to adjust the constraints to achieve feasibility with the minimum effort. To this end, phase II compute such a set of optimal adjustments to be applied

on constraints via solving a multi-objective optimization problem.

It was pointed out in [180] that any method based on a weighted sum of elastic variables to determine the constraint shifts can arrive at only a limited set of possible solutions: those that appear at the corner points of the solution space. A goal programming approach, on the other hand, allows solutions that arrive at any point on the efficient frontier (the Pareto optimal set), giving the modeller a significantly larger set of possible constraint shifts that provides a feasible solution [43].

For the convenience of analysis, we restore to the formulation (3.2) with both equalities and inequalities in the sequel. For an infeasible problem (3.2) having inequality constraints $g_j(x)$, $j \in \mathcal{I}$, the constraint shifting problem can be formulated as the following multi-objective optimization problem:

$$\begin{aligned} \min \quad & (y_1, y_2, \cdots, y_{n_\mathcal{I}}) \\ s.t. \quad & g_j(x) - y_j \leq 0 \\ & y_j \geq 0, \ j = 1, \cdots, n_\mathcal{I}, \end{aligned} \tag{3.14}$$

where, y_j are non-negative relaxation variables representing the adjustments to be applied on the constraints in order to regain feasibility.

The l_∞-norm based method is first used to solve the multi-objective optimization problem (3.14). The objective function based on the l_∞-norm is:

$$\begin{aligned} \min_{x,y,z} \quad & z = \max_{j=1,\cdots,n_\mathcal{I}} |w_j y_j| \\ s.t. \quad & g_j(x) - y_j \leq 0, \ j = 1, \cdots, n_\mathcal{I}, \end{aligned} \tag{3.15}$$

where, $w_j \geq 0$ and $\sum_{j=1}^{n_\mathcal{I}} w_j = 1$, $j = 1, \cdots, n_\mathcal{I}$ are the weights imposed on the adjustments of the constraints. The optimization problem (3.15) can be imple-

mented as

$$\begin{aligned} \min_{x,y,z} \quad & z \\ s.t. \quad & w_j y_j \leq z, \ j = 1, \cdots, n_{\mathcal{I}} \\ & g_j(x) - y_j \leq 0, \ j = 1, \cdots, n_{\mathcal{I}} \\ & y_j \geq 0, \ j = 1, \cdots, n_{\mathcal{I}}. \end{aligned} \quad . \tag{3.16}$$

The absolute value is not needed because both w and y are non-negative. An optimal solution to the l_∞-norm optimization problem (3.15) is denoted as $(\tilde{x}^*, \tilde{y}^*)$. This is a weakly efficient solution to the multi-objective optimization problem (3.14).

Then, the solution $(\tilde{x}^*, \tilde{y}^*)$ is improved by solving a second optimization problem, which is a weighted relaxation program formulated as follows:

$$\begin{aligned} \min_{x,y} \quad & \sum_{j=1}^{n_{\mathcal{I}}} w_j y_j \\ s.t. \quad & g_j(x) - y_j \leq 0, \ j = 1, \cdots, n_{\mathcal{I}} \\ & 0 \leq y_j \leq \tilde{y}_j^*, \ j = 1, \cdots, n_{\mathcal{I}}. \end{aligned} \tag{3.17}$$

An optimal solution to the optimization problem (3.17), denoted as (x^*, y^*) is the final solution to the task of achieving feasibility with least constraint shifts.

The combination of l_∞-norm optimization with subsequent tightening allows any solution on the efficient frontier of the multi-objective program (3.14) to be reached by adjusting the weights w. This provides the modeller with a vastly increased set of possible ways to shift the constraints to regain feasibility [180]. The choice of weights is usually application oriented, which generally represent priorities of the constraints. More specifically, large weights are usually imposed on high-priority constraints, while small weights on low-priority constraints.

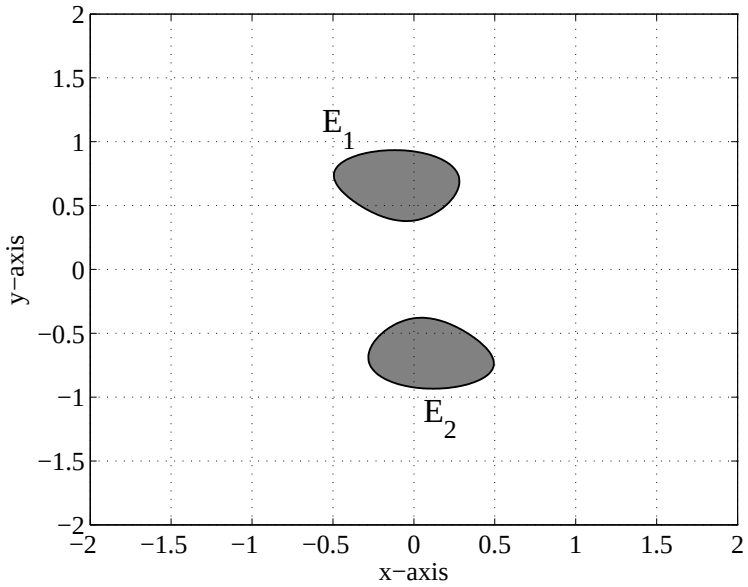


Figure 3.6: The problem (3.18) has two disconnected feasible components, E_1 and E_2.

3.3.3 Illustrative examples

In this section, we present two examples for the purpose of illustrating the proposed TRUST-TECH based methods for feasibility analysis.

Feasibility computation example

We first consider the following feasibility computation problem:

$$g(x,y) \;=\; 4x^2 - 2.4x^4 + \frac{1}{3}x^7 + xy - 4y^2 + 4y^4 + \frac{1}{2} \leq 0. \tag{3.18}$$

This problem has two disconnected feasible components, as shown in Fig. 3.6.

To carry out the TRUST-TECH based feasibility analysis, a slack item s^2 is first added to (3.18) to transform it into an equality constraint in the following

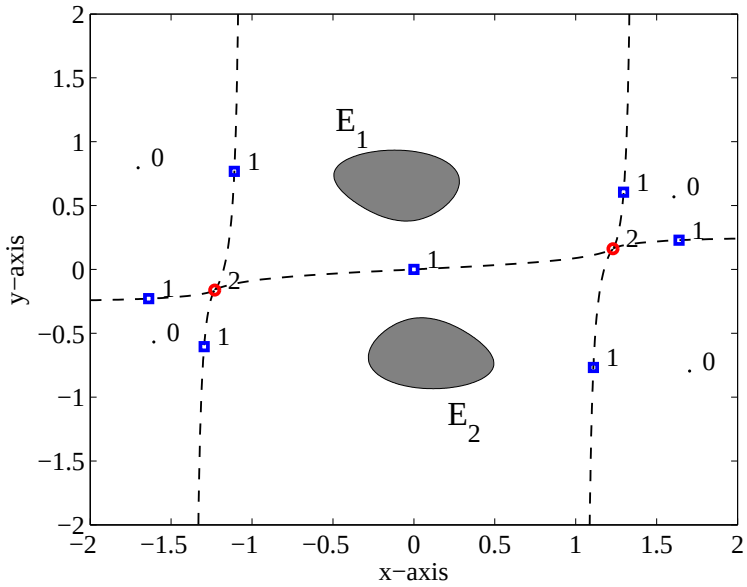


Figure 3.7: The QGS (3.20) has two stable equilibrium manifolds, E_1 and E_2 other thirteen equilibrium points (0: stable; 1 and 2: unstable). Dashed curves are stability boundary.

equivalent form:

$$h(x,y,s) \;=\; 4x^2 - 2.4x^4 + \frac{1}{3}x^7 + xy - 4y^2 + 4y^4 + \frac{1}{2} + s^2 = 0. \tag{3.19}$$

Then, the following QGS is constructed

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{s} \end{bmatrix} \;=\; -\nabla^T h(x,y,s)h(x,y,s) \;=\; \begin{bmatrix} -h(x,y,s)\frac{\partial}{\partial x}h(x,y,s) \\ -h(x,y,s)\frac{\partial}{\partial y}h(x,y,s) \\ -h(x,y,s)\frac{\partial}{\partial z}h(x,y,s) \end{bmatrix}. \tag{3.20}$$

We first choose any initial guess, say $\mathbf{x}_0 = (-1.5, -1.5, 10.0)$, which is an infeasible point because $g(\mathbf{x}_0) = 16.166 > 0$. Following the system trajectory (S_1), an SEP of the QGS (3.20), $E_0 = (-1.61, -0.57, 0.0)$ is obtained. However, it is not a feasible point because $g(E_0) = 2.604$.

Next, starting from E_0, a TRUST-TECH search (S_2) is performed and an exit point $\mathbf{x}_e = (-1.21, -0.41, 0.0)$ is located in the eigenvector search direction $d =$

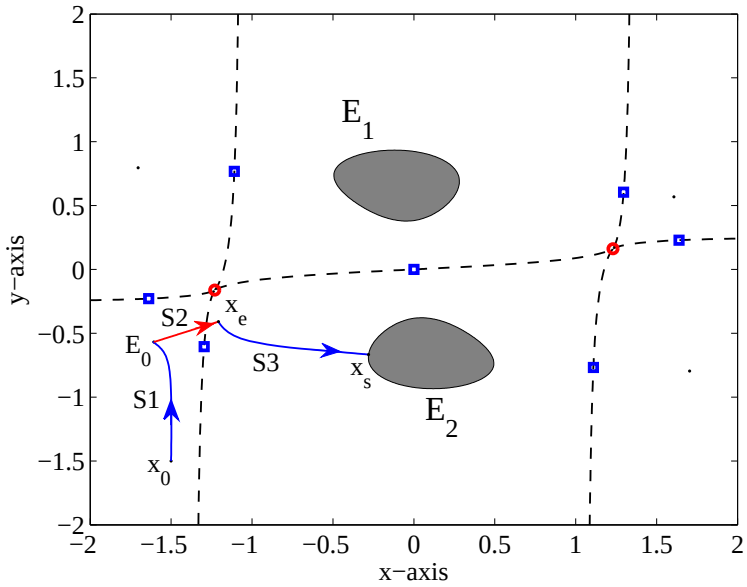


Figure 3.8: The process to solve (3.18). x_0: starting point; E_0: stable equilibrium point (S_1: trajectory); x_e: exit point; x_s: feasible point; S_1, S_3: system trajectory; S_2: search direction.

$(0.93, 0.37, 0.0)$. Starting from $\mathbf{x}_e$, the stable equilibrium manifold E_2 is reached at $\mathbf{x}_s = (-0.28, -0.67, 0.0)$ by following the system trajectory S_3. Since $g(\mathbf{x}_s) = -0.001$, it is indeed a feasible point to the problem (3.18).

This example shows that, if a local method is used solely starting from a poor initial point (such as x_0 in this example), a truly feasible problem can be falsely reported as infeasible because the local method can only reach an infeasible solution (E_0 in this example) and can not escape from this solution. The TRUST-TECH based method is able to eliminate such false negative results by systematically locating multiple stable equilibrium manifolds of the QGS (3.20), among which the truly feasible component can be identified.

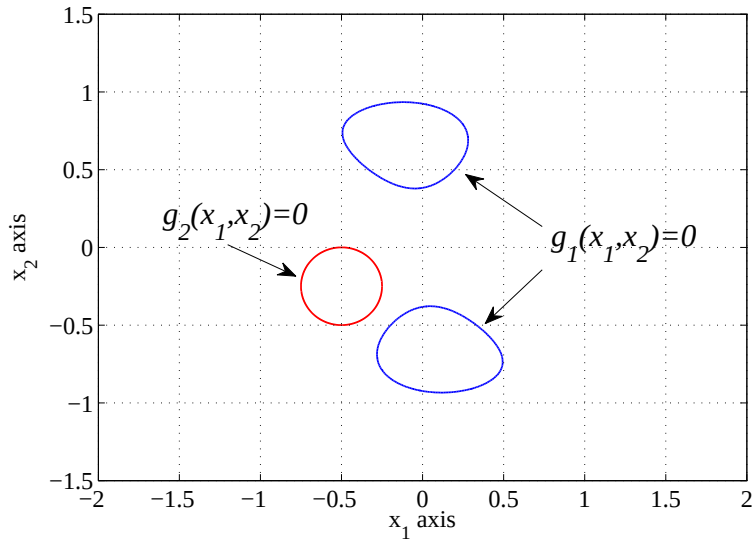


Figure 3.9: The problem (3.21) is infeasible.

Feasibility restoration example

We now consider the following problem for illustration of the TRUST-TECH based method for feasibility restoration:

$$\begin{aligned} g_1(x_1, x_2) &= 4x_1^2 - 2.4x_1^4 + \frac{1}{3}x_1^7 + x_1x_2 - 4x_2^2 + 4x_2^4 + \frac{1}{2} \leq 0 \\ g_2(x_1, x_2) &= (x_1 + \frac{1}{2})^2 + (x_2 + \frac{1}{4})^2 - \frac{1}{5} \leq 0 \end{aligned} \quad . \tag{3.21}$$

As illustrated in Fig. 3.9, this problem is infeasible since there is no single which can satisfy the two inequalities simultaneously.

By adding slack items s_1^2 and s_2^2, inequality constraints in (3.21) are first transformed to equality constraints in the following equivalent form:

$$H(x_1, x_2, s_1, s_2) = \begin{pmatrix} 4x_1^2 - 2.4x_1^4 + \frac{1}{3}x_1^7 + x_1x_2 - 4x_2^2 + 4x_2^4 + \frac{1}{2} + s_1^2 \\ (x_1 + \frac{1}{2})^2 + (x_2 + \frac{1}{4})^2 - \frac{1}{16} + s_2^2 \end{pmatrix} = 0. \tag{3.22}$$

Table 3.1: Stable equilibrium points found by TRUST-TECH in the QGS (3.23).

SEP	x_1	x_2	s_1	s_2	$g_1(x_1, x_2)$	$g_2(x_1, x_2)$	$\frac{1}{2}\|H(x_1, x_2)\|^2$
1	-0.197	0.334	0	0	0.190	0.370	0.087
2	-0.208	-0.500	0	0	0.024	0.085	0.004

Then, the following QGS is constructed

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{s}_1 \\ \dot{s}_2 \end{bmatrix} = -\nabla^T H(x_1, x_2, s_1, s_2) \cdot H(x_1, x_2, s_1, s_2). \tag{3.23}$$

There are two stable equilibrium points found by TRUST-TECH search in the QGS (3.23), as summarized in Table 3.3.3.

Among the two found SEPs, the second SEP $(-0.208, -0.500)$ results in lower QGS energy function value. Therefore, this SEP is used as the initial point in the multi-objective optimization problem (3.14). The weights are specified as $w_1 = 0.75$ and $w_2 = 0.25$; in other words, we prefer more adjustments over the constraint $g_2(x_1, x_2)$ than over the constraint $g_1(x_1, x_2)$. As the output of the optimization problem (3.14), the adjustments required on the two constraints are $y_1 = 0.028$ and $y_2 = 0.084$. The resulting feasible point is $(x_1, x_2) = (-0.209, -0.499)$, as shown in Fig. 3.10.

The change of the adjustments to be applied to the constraints for varying weights is also studied. The SEP $(-0.208, -0.500)$ is used as the initial point in the multi-objective optimization problem (3.14). We increase the weight w_1 on $g_1(x_1, x_2)$ from 0.05 to 0.95 (accordingly, w_2 on $g_2(x_1, x_2)$ will be decreased from

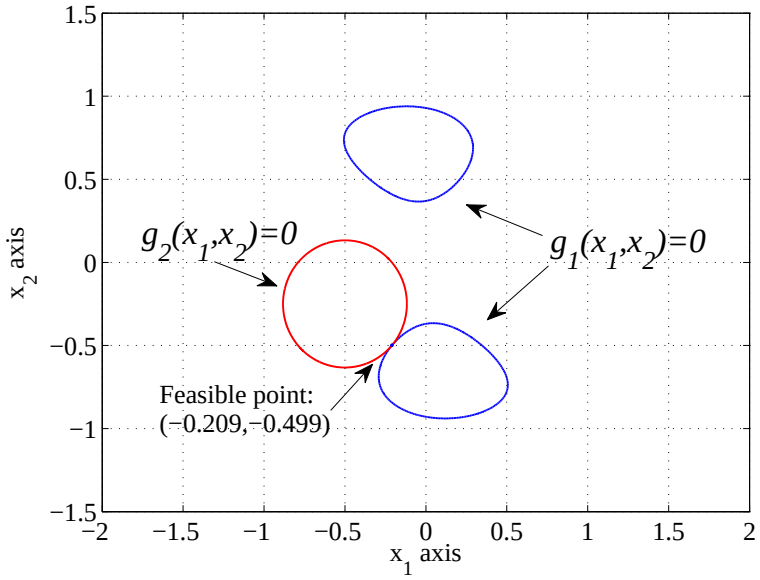


Figure 3.10: Using $(-0.208, -0.500)$ as the initial point, the optimal adjustment is achieved.

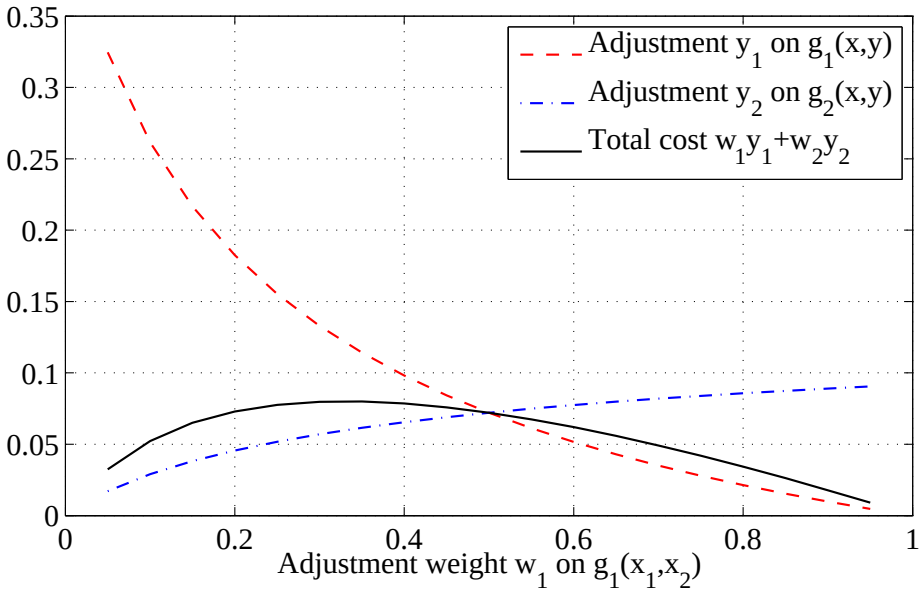


Figure 3.11: Using $(-0.197, 0.334)$ as the initial point, change of the adjustments with varying weights.

0.95 to 0.05). Fig. 3.11 shows the adjustments y_1 and y_2 under different weights. The change of the total adjustment cost $z = w_1y_1 + w_2y_2$ is also demonstrated.

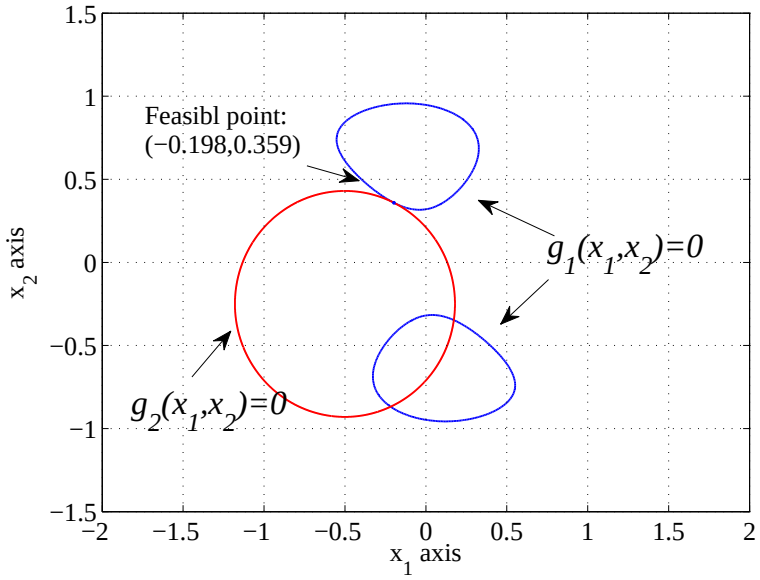


Figure 3.12: Using $(-0.197, 0.334)$ as the initial point, an over-adjustment is obtained.

Now we demonstrate what will happen if the initial point chosen for the multi-objective optimization problem (3.14) is not the global minimum violation point. We now use the SEP $(-0.197, 0.334)$ as the initial point in solving the optimization problem (3.14) with the weights keep the same, being $w_1 = 0.75$ and $w_2 = 0.25$. As a result of the computation, the obtained adjustments are $y_1 = 0.133$ and $y_2 = 0.400$ and the feasible point is $(-0.198, 0.359)$, as shown in Fig. 3.12. Compared with the situation using $(-0.208, -0.500)$ as the initial point, the adjustments are much larger. Apparently, this restoration is local optimal and the constraints are over-adjusted. In fact, besides the feasible point $(-0.198, 0.359)$ detected by the optimization problem (3.14), there emerges another isolated feasible component after adjusting the constraints, which has been overlooked because of the poor initial point (Fig. 3.12).

3.4 Applications to Power System Analysis

In this section, the TRUST-TECH based feasibility analysis methods are applied to solve practical problems in power system analysis, including power flow computation and feasibility restoration for optimal power flow (OPF) problems. As an efficient numerical implementation, the pseudo-transient continuation method is first introduced, which is used in the numerical implementation of the proposed methods to achieve a fast convergence.

3.4.1 The pseudo transient continuation method

Locating a stable equilibrium manifold of the QGS (3.9), i.e., computing a feasible solution to the feasibility problem (3.2), from an initial point usually needs to follow the system trajectory. Generally, an *ordinary differential equation (ODE)* solver can be used to exactly follow the system trajectory until a stable equilibrium manifold is reached. However, an exact ODE solver can be computationally inefficient, in particular for large-scaled systems which is not uncommon in power system applications. The pseudo-transient continuation method is used in the numerical implementation for power flow analysis.

Newton's method are a classical algorithm for solving the system of nonlinear equations

$$H(x) = 0 \tag{3.24}$$

with quadratic convergence if the initial point is close enough to a solution. However, solving the system of linear equations (the Newton equations)

$$\nabla^T H(x) \cdot \Delta x = -H(x) \tag{3.25}$$

at each stage can be expensive if the number of variables is large and may not be justified when the initial point is far from a solution.

The pseudo-transient continuation method [46, 93] is another way to implement Newton-type methods. This method was originally designed as a method for finding steady-state solutions to time-dependent differential equations without computing a fully time-accurate solution.

In the context of optimization, one would integrate (3.9) numerically, managing the time step in a way that, while maintaining stability, would increase as rapidly as possible in order to make the transition to Newton's method near the solution. One way to do this is the iteration

$$x_{n+1} = x_n - (\sigma_n^{-1} I + J_H(x_n))^{-1} H(x), \tag{3.26}$$

where, $J_H(x) = H'(x)$ is the system Jacobian. One common way to update the time step δ_n is the switched evolution relaxation method:

$$\delta_{n+1} = \delta_n \frac{||H(x_n)||}{||H(x_{n+1})||}. \tag{3.27}$$

Hence, the time step will be increased as the stable equilibrium points of the system (3.9) are approached, thus the objective of rapid convergence near a solution can be achieved.

3.4.2 Power flow computation

The first problem to be solved is power flow computation, which is one of the most fundamental tasks in power system studies. After half a century of intensive research, development and practical application, the power flow com-

putation remains difficult [87]. Depending on the system's characteristics, conventional power flow methods may fail, even if there is a feasible operating point [13]

In this section, the TRUST-TECH based feasibility computation method is used for power flow analysis. Key features of the proposed method are twofold, including:

1) *Non-divergent:* If there exists a power flow solution, the TRUST-TECH based method can always compute the solution.
2) *Unsolvability detection:* If the system does not possess a power flow solution, instead of divergence, the TRUST-TECH based method will converge to a point with useful information regarding the unsolvability.

The power flow formulations

The power flow computation is to determine a set of voltage settings at each bus of the system such that a balance between the power generation and demand can be achieved. Since the voltage at each bus is a complex number, the power flow analysis can be formulated in two coordinate systems, that is, the polar coordinate system and the rectangular coordinate system. In the polar power flow formulation, the complex voltage phaser at each bus is represented using polar coordinates

$$V_i = |V_i| \angle \theta_i. \tag{3.28}$$

For the rectangular power flow formulation, the voltage phaser at each bus is represented using rectangular coordinates

$$V_i = e_i + j f_i. \tag{3.29}$$

Using polar coordinates, the basic power flow equation for an n_B-bus power system can be formulated as the following system of nonlinear equations representing power balances at all the buses in the system:

$$\begin{aligned} P^i(\theta, |V|) + P_D^i - P_G^i &= 0,\ i = 1, \cdots, n_B \\ Q^i(\theta, |V|) + Q_D^i - Q_G^i &= 0,\ i = 1, \cdots, n_B \end{aligned}, \tag{3.30}$$

where,

- $\theta = (\theta_1, \cdots, \theta_{n_B})$ and $|V| = (|V_1|, \cdots, |V_{n_B}|)$ are the vectors of voltage phase angles and magnitudes at all the buses in the network, respectively;
- P_D^i and Q_D^i are the real and reactive power demands at the i-th bus, respectively;
- P_G^i and Q_G^i are the real and reactive power generations at the i-th bus, respectively;
- $P^i(\theta, |V|)$ and $Q^i(\theta, |V|)$ are the real and reactive power injection into the network at the i-th bus, and are formulated as the follows:

$$\begin{aligned} P^i(\theta, |V|) &= |V_i| \sum_{j \in \mathcal{N}_i} \left\{ |V_j| (G_{ij} \cos(\theta_i - \theta_j) + B_{ij} \sin(\theta_i - \theta_j)) \right\} \\ Q^i(\theta, |V|) &= |V_i| \sum_{j \in \mathcal{N}_i} \left\{ |V_j| (G_{ij} \sin(\theta_i - \theta_j) - B_{ij} \cos(\theta_i - \theta_j)) \right\} \end{aligned}, \tag{3.31}$$

  where $Y_{ij} = G_{ij} + jB_{ij}$ is the complex admittance between buses i and j, and $\mathcal{N}_i$ denotes the set of neighbourhood buses which are connected to bus i, including itself.

For the rectangular formulation, the power flow equations contain real and reactive power balances and voltage set-point equations of the following forms:

$$\begin{aligned} P^i(V_e, V_f) + P_D^i - P_G^i &= 0,\ i = 1, \cdots, n_B \\ Q^i(V_e, V_f) + Q_D^i - Q_G^i &= 0,\ i = 1, \cdots, n_B \end{aligned}. \tag{3.32}$$

where

- $V_e = (e_1, \cdots, e_{n_B})$ and $V_f = (f_1, \cdots, f_{n_B})$ are the vectors real and imaginary voltage magnitudes;
- $P^i(V_e, V_f)$ and $Q^i(V_e, V_f)$ are the real and reactive power injections at the i-th bus formulated as follows:

$$\begin{aligned} P^i(V_e, V_f) &= \sum_{j \in \mathcal{N}_i} \Big\{ e_i(G_{ij}e_j - B_{ij}f_j) + f_i(G_{ij}f_j + B_{ij}e_j) \Big\} \\ P^i(V_e, V_f) &= \sum_{j \in \mathcal{N}_i} \Big\{ f_i(G_{ij}e_j - B_{ij}f_j) - e_i(G_{ij}f_j + B_{ij}e_j) \Big\}. \\ V_i(V_e, V_f) &= e_i^2 + f_i^2 - |V_i|^2 = 0 \end{aligned} \tag{3.33}$$

It should be noted that the rectangular formulation uses an extra equation at each PV buses in the system, due to the need to maintain the specified voltage magnitude. As a result, the rectangular formulation (3.32) has a larger equation and variable count relative to the polar formulation (3.30), with different equal to the number of PV buses in the system.

Therefore, the task of power flow analysis is to compute a vector of state variables $x = (\theta, |V|)$ in polar coordinates to satisfy the polar power flow equations (3.30), or $x = (V_e, V_f)$ in the rectangular coordinates to satisfy the rectangular power flow equations (3.32).

Numerical study

The TRUST-TECH based method is performed to compute power flow solutions on benchmark power systems with the number of buses ranging from 6 to 2383. In this numerical study, the power flow problem is solved in its polar formulation (3.30). Moreover, in order to trace the power flow computation

from solvable to unsolvable conditions, the load demands at each bus are consistently increased from the base values (P_D^0 and Q_D^0) with an increasing loading parameter λ:

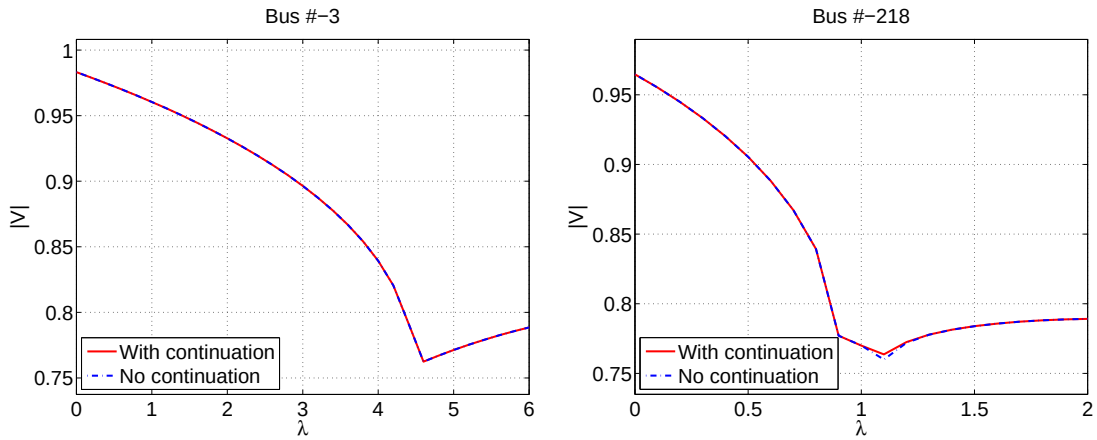
$$\begin{aligned} P_D &= (1+\lambda)P_D^0 \\ Q_D &= (1+\lambda)Q_D^0 \end{aligned}. \tag{3.34}$$

Two computation schemes are considered and compared in tracing power flow solutions with changing loading conditions. The first scheme is a naive scheme, where the power flow analysis is carried out with a fixed initial condition regardless of the change in load condition. The other scheme utilizes a continuation strategy, where the power flow solution obtained at the previous iteration is used as the initial condition at the current iteration. Such continuation strategy has been widely used in continuation power flow analysis for voltage stability analysis [34, 110].

The computational results on 30-bus and 2383-bus systems are shown in Fig. 3.13. In this figure, the power flow solutions computed with the two strategies are compared. In each case, the load is increased until the power flow problem (3.30) is not feasible. The change of the QGS energy value (3.11) along the solution process is also presented.

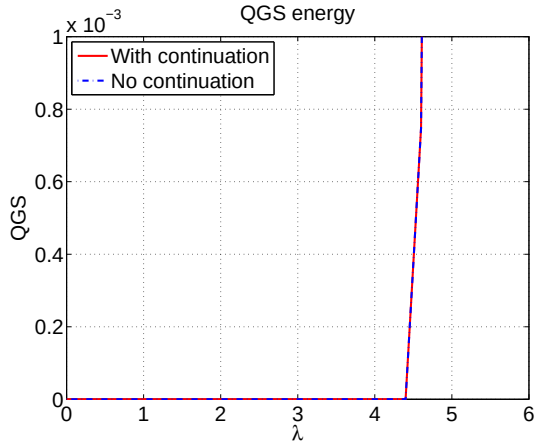
We have the following observations based on the results:

- The two schemes generate almost identical results, in terms of both the change of the solution and the change of the QGS energy. A neglectable deviation can be observed on the 2383-bus system when the loading condition λ becomes larger than 1.0.
- Even without using continuation, the TRUST-TECH based method can

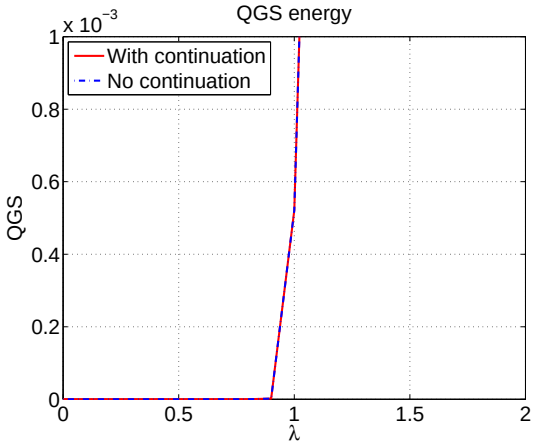


(a) 30-bus system, bus-3

(b) 2383-bus system, bus-218



(c) 30-bus system, QGS energy



(d) 2383-bus system, QGS energy

Figure 3.13: This figure compares power flow solutions computed with different strategies and the change of QGS energy at the computed solutions. The solid curve represents the power flows computed without using continuation, while the dashed curve represents the power flows computed using continuation. Solutions by the two schemes match to each other very well. The abrupt change of QGS energy indicates the unsolvability of the power flow.

still follow exactly the solution curve all the way to severe loading conditions where conventional power flow solvers will fail to give a solution or just diverge.

- The QGS energy provides an indicator for the solvability of the power

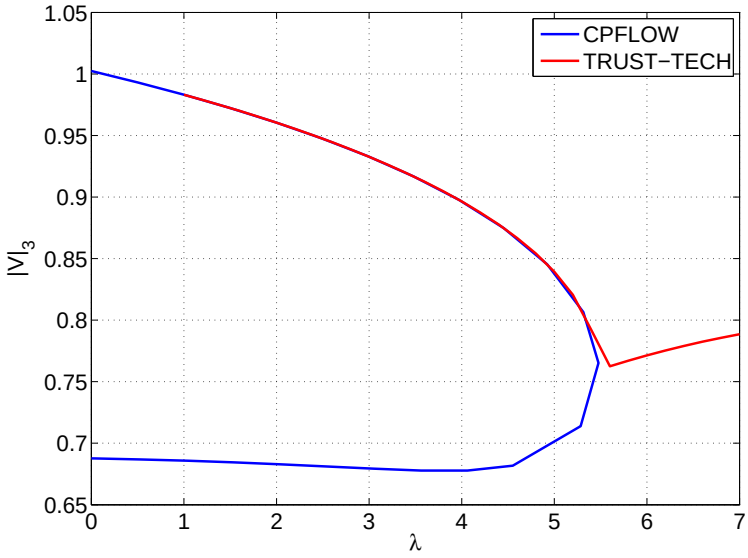


Figure 3.14: Comparison between the P-V curves on the 30-bus system obtained using the continuation power flow method [110] and the TRUST-TECH based method without using the continuation scheme.

flow problem (3.30). When the power flow problem (3.30) is solvable, the attained QGS energy will always be zero, indicating the solvability. On the other hand, once the power flow problem (3.30) becomes unsolvable, an abrupt increase of the QGS energy can be observed.

The computation result is also compared to the solution obtained by a continuation power flow solver called CPFLOW [110]. The result by the TRUST-TECH based method without continuation is considered in this comparison. Fig. 3.14 shows the P-V curves at the bus #3 in the 30-bus system. The validity the TRUST-TECH based method is demonstrated through the almost exact match between the results (before the nose point is reached, beyond which the power flow problem (3.30) becomes unsolvable).

3.4.3 Feasibility restoration for optimal power flow

The second problem to be studied is the optimal power flow (OPF) problem. Since the early 1960s, the OPF problem has been one of the most widely studied topics in power system analysis and computation [23, 51]. This problem is relevant in power system operations, scheduling, and planning [159, 125]. The main objective of the OPF problem is to determine the optimal steady-state operation of an electric power system while satisfying engineering and economic constraints. With the structural deregulation of electric power systems, OPF is becoming a basic tool in the power market.

Optimal power flow

Without loss of generality, a typical OPF model formulated in poor coordinates is considered in this section:

$$\begin{aligned}
\min \quad & f(V,\theta,P^G,Q^G) \\
s.t. \quad & P_i(V,\theta)+P_i^D-P_i^G \;=\; 0,\; i=1,\cdots,n_B \\
& Q_i(V,\theta)+Q_i^D-Q_i^G \;=\; 0,\; i=1,\cdots,n_B \\
& S_{ij}(V,\theta)\le \overline{S}_{ij},\; (i,j)\in\mathcal{L} \\
& S_{ji}(V,\theta)\le \overline{S}_{ij},\; (i,j)\in\mathcal{L} \\
& \underline{V}_i \le V_i \le \overline{V}_i,\; i=1,\cdots,n_B \\
& \underline{P}_j^G \le P_j^G \le \overline{P}_j^G,\; j=1,\cdots,n_G \\
& \underline{Q}_j^G \le Q_j^G \le \overline{Q}_j^G,\; j=1,\cdots,n_G
\end{aligned} \quad , \tag{3.35}$$

where, $P^G=\{P_j^G\}$ and $Q^G=\{Q_j^G\}$, $j=1,\cdots,n_G$, are the real and reactive power outputs of generators, respectively; $V=\{V_i\}$ and $\theta=\{\theta_i\}$, $i=1,\cdots,n_B$, are the magnitude and phase angle the bus voltages, respectively; S_{ij} and S_{ji} are the power flow through the (i,j)-th line measured at the from-end and the to-end of

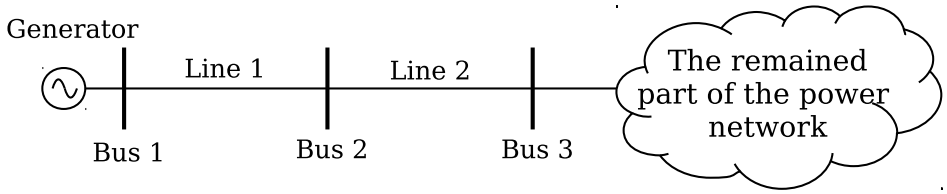


Figure 3.15: Illustration of a power system. The optimal power flow problem will be infeasible if the lower bound of the generator's output is larger than the thermal limit imposed on Line 1 or Line 2.

the line, respectively; P^D, Q^D are the active and reactive load powers of all buses; $\mathcal{L}$ is the set of branches in the system; $\underline{x}$ means the lower limit of x, $\overline{x}$ means the upper limit of x. In the OPF problem, the objective function $f(x)$ can be fuel cost generation, active and/or reactive power transmission loss, reactive power reserve margin, security margin index, emission, and environmental index.

An infeasible OPF can occur if there are improper limits imposed on the optimization variables, which may caused by data errors. A typical cause of infeasible OPFs can be illustrated with help of the power system shown in Fig. 3.15. In this power system, a generator is connected to bus 1, which is connected to bus 2 via line 1. Line 2 connects buses 2 and 3, and bus 3 is connected to the remained part of the power network. Suppose there are no loads attached to the three buses. Therefore, all power produced by the generator, in excess of the power losses over lines 1 and 2, will be injected into the network. Now assume that the lower limit of the power produced by the generator is larger than the maximum allowable power flow (the thermal limit) through line 1 (or line 2). Under such configuration, all allowable power output from the generation will be larger than the thermal limit imposed on line 1(or line 2). Specifically, the line flow $S_{12} \approx P_1^G$ (or $S_{23} \approx P_1^G$) will be always larger than its upper bound $\overline{S}_{12}$ (or $\overline{S}_{13}$) for all $\underline{P}_1^G \leq P_1^G \leq \overline{P}_1^G$. Therefore, the OPF problem is infeasible since there is

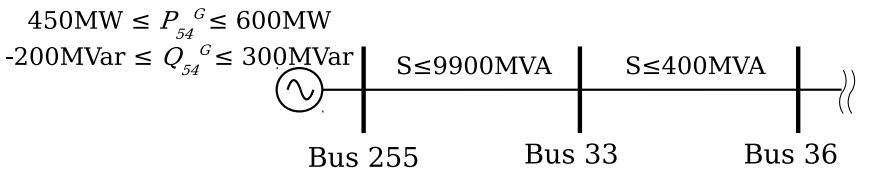
no suitable value of P_1^G to satisfy both the generation bounds and thermal limits simultaneously.

A method to check feasibility of the OPF problem and to restore feasibility if an infeasible OPF problem is detected can be a beneficial step before the OPF solver is applied. Indeed, it has been shown in [38] that obtaining a feasible solution beforehand can not only improve the overall computational efficiency but also remedy divergence of the solver if a poor initial condition is used for a feasible OPF problem. In addition, if the OPF problem is infeasible, a lot of CPU time can be wasted by the solver and it generally can not output useful information regarding the underlying cause of its divergence. To this end, the TRUST-TECH based method developed in this chapter is used for feasibility computation and restoration of OPF problems.

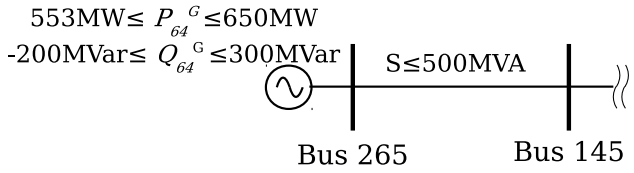
Numerical study

In the numerical study, the TRUST-TECH based method is carried out on a 300-bus power system. This system has 69 generators and 411 lines. The outputs of the 54-th generator attached to bus 255 are $450MW \leq P_{54}^G \leq 600MW$ and $-200MVar \leq Q_{54}^G \leq 200MVar$, while the maximum allowable flow through the line connecting buses 33 and 36 is set to be $400MVA$ (Fig. 3.16(a)). Similarly, the outputs of the 64-th generator are $553MW \leq P_{64}^G \leq 650MW$ and $-200MVar \leq Q_{64}^G \leq 300MVar$, while the thermal limit of the line connecting buses 265 and 145 is $500MVA$ (Fig. 3.16(b)).

In stage I computation of the TRUST-TECH based method, the QGS is constructed based on the OPF constraints and the pseudo-transient continuation



(a) The generator attached to bus 255



(b) The generator attached to bus 265

Figure 3.16: Two parts of the 300-bus power system with improper bounds on generators' output which cause infeasibility of the OPF problem.

method is used to compute stable equilibrium manifolds in the QGS. The minimum violation point obtained in stage I has an QGS energy value of 0.212; hence, this OPF problem is indeed infeasible.

Using the minimum violation point obtained in stage I as the initial point, stage II computes the optimal shifts on the constraints to restore feasibility for the OPF problem. In this numerical study, only the (lower and upper) bounds of generators' outputs are allowed to be adjusted. An interior point solver is used to solve the involved optimization problems (3.16) and (3.17). All generation bounds are treated equally, that is, the weights imposed on all generation bounds are the same. In the solution of stage II, the computed shifts on lower bounds of the 54-th and 64-th generators' real power output are $50.00032MW$ and $53.00048MW$, respectively. All shifts on other (real and reactive) generation (lower and upper) bounds are smaller than $2 \times 10^{-4} MW/MVar$. In other words, the bounds of the real power generation of the 54-th generator should be adjusted to $400MW \leq P_{54}^{G} \leq 600MW$ and that of the 64-th generator should be

adjusted to $500MW \leq P_{64}^{G} \leq 650MW$.

3.5 Summary

In this chapter, TRUST-TECH based methods are developed for feasibility analysis. Following the spirit in the TRUST-TECH methodology, the feasibility analysis problem is solved with the aid of a particular nonlinear dynamical system. Specifically, finding feasible components satisfying the imposed constraints is accomplished via finding all the stable equilibrium manifolds in the constructed quotient gradient system. For feasibility restoration, a two-phase TRUST-TECH based method is developed. It first finds the global minimum-violation component via systematically finding all local minimum-violation components via finding all the stable equilibrium manifolds in the same quotient gradient system; then it computes the optimal strategy of adjusting the constraints to achieve feasibility. Indeed, the TRUST-TECH methodology has introduced a unified framework for analysing feasibility and infeasibility for nonlinear problems. The proposed methods are applied to compute power flow solutions and to restore feasibility for infeasible optimal power flow problems with promising results.

From a computational viewpoint, however, it may well be due to the lack of computation methods for computing the complete unstable manifolds, the TRUST-TECH-based methods can only compute multiple feasible components (local minimum-violation components for infeasible problems), instead of all of the feasible components (local minimum-violation components for infeasible problems) for general feasibility analysis problems.

CHAPTER 4

ENHANCED TRUST-TECH BASED METHODS FOR CONSTRAINED NONLINEAR OPTIMIZATION

4.1 Introduction

In this chapter, we mainly study TRUST-TECH based methods for solving the following constrained global optimization problem:

$$\begin{aligned} \min \quad & f(x) \\ s.t. \quad & h_i(x) = 0,\ i \in \mathcal{E} = \{1, \cdots, n_{\mathcal{E}}\} \\ & g_j(x) \le 0,\ j \in \mathcal{I} = \{1, \cdots, n_{\mathcal{I}}\} \end{aligned} \quad , \tag{4.1}$$

where, the objective function $f(x)$, the nonlinear equality constraints $h_i(x)$, $i \in \mathcal{E}$ and the nonlinear inequality constraints $g_j(x)$, $j \in \mathcal{I}$ are all twice differentiable, that is, they all belong to $C^2 : R^n \to R$.

The *Lagrangian function* $L : R^{n+n_{\mathcal{E}}+n_{\mathcal{I}}} \to R$ of (4.1) is defined by

$$L(x, \lambda, \mu) \ = \ f(x) + y^T h(x) + z^T g_j(x). \tag{4.2}$$

where, $h = (h_1, \cdots, h_{n\mathcal{E}})^T$ and $g = (g_1, \cdots, g_{n\mathcal{I}})^T$, y and z are Lagrange multipliers for the equality and inequality constraints, respectively. We call $\bar{x}$ a *critical point* of (4.1) (with Lagrange multipliers $(\bar{y}, \bar{z})$) if

$$\begin{aligned} \nabla_x L(\bar{x}, \bar{y}, \bar{z}) \ &= \ \nabla f(\bar{x}) + \nabla^T h(\bar{x})\bar{y} + \nabla^T g(\bar{x})\bar{z} \ = \ 0 \\ h(\bar{x}) \ &= \ 0 \\ g(\bar{x}) \ &\le \ 0 \end{aligned} \quad . \tag{4.3}$$

The conditions (4.3) are known as *Karush-Kuhn-Tucker (KKT)* conditions. The constraint set of (4.1) defines the following feasible set:

$$S \ = \{x \in R^n : h_i(x) = 0, i \in \mathcal{E},\ g_j(x) \le 0, j \in \mathcal{I}\}, \tag{4.4}$$

which can be any closed subset of R^n, and its structure can be very complex. The constraint set S is usually non-convex and disconnected; i.e. it is composed of several disjoint and path-connected feasible regions. The task of locating each connected feasible region of the set S is in itself a difficult one. A reasonable local structure of the constraint set can be obtained under the following assumptions, which is generically true [89]:

1. *(Regularity)* At each $x \in S$, the set of vectors

$$\{\nabla h_i(x), i \in \mathcal{E}\}\{\nabla g_j(x), j \in \mathcal{I}_\alpha(x)\} \tag{4.5}$$

   are linearly independent where

$$\mathcal{I}_\alpha(x) \;=\; \{j \in \mathcal{I} : g_j(x) = 0\} \tag{4.6}$$

   is the index set for the active constraints. This condition is also known as *linear independence constraint qualification (LICQ)*

2. *(Nondegeneracy)* At each critical point, $\bar{x} \in S$, we have

$$d^T \nabla^2_{xx} L(\bar{x}, \bar{\lambda}, \bar{\mu}) d \neq 0, \;\; \forall d \neq 0 \tag{4.7}$$

   where, d satisfies

$$\begin{array}{l} \nabla^T h_i(\bar{x}) d = 0, \forall i \in \mathcal{E} \\ \nabla^T g_j(\bar{x}) d = 0, \forall j \in \mathcal{I}_\alpha(\bar{x}) \end{array}. \tag{4.8}$$

3. *(Strict Complementary)*

$$\bar{\mu}_j \neq 0, \;\; \forall j \in \mathcal{I}_\alpha(\bar{x}). \tag{4.9}$$

4. *(Finiteness and Separating Property)* The objective function f has finitely many critical points in S at which it attains different values of f.

Without loss of generality, we consider the following optimization problem with equality constraints:

$$\begin{array}{ll} \min & f(x) \\ s.t. & h_i(x) = 0, i \in \mathcal{E} = \{1, \cdots, m\} \end{array} . \tag{4.10}$$

We define the following equality vector $H(x) := (h_1(x), \cdots, h_m(x))^T$. By applying the regularity condition and Sard's theorem [149], it can be shown that the constraint set (or the feasible set or region) defined by

$$M = \{x \in R^n : H(x) := (h_1(x), \cdots, h_m(x))^T = 0\} \tag{4.11}$$

is a smooth manifold. In general, the extremely complicated constraint set M can be decomposed into several disjoint path-connected feasible components.

One challenge in solving the nonlinear optimization problem (4.10) is that the feasible set M can be very complex and contain multiple feasible components. As a result, the existence of multiple local optimal solutions to the optimization problem (4.1) can not only due to the nonlinearity of the objective function, but due to the complexity of the feasible region. The great majority of existing numerical methods/techniques and heuristics for solving constrained nonlinear optimization problems are usually globally convergent and can only verify the existence of feasible solutions after a feasible solution is found. In other words, when these methods fail to acquire a feasible solution, they usually can not verify whether such infeasibility is local or global.

This section develops TRUST-TECH based methods for computing multiple local optimal solutions to the constrained optimization problem (4.1). Specifically, two methods are proposed. The first method is derived from the concept of interior point methods (IPMs) and is named *TRUST-TECH based interior point method (TT-IPM)*. In this method, a TRUST-TECH search is directly carried out in

the constructed KKT gradient system to search for multiple stable equilibrium points with zero energy, each of which corresponds to a local optimal solution to the optimization problem (4.1). The second method is a two-staged method and is called *reduced projected gradient method*. The first stage of this method is to locate multiple feasible components by computing multiple stable equilibrium manifolds in an associated *quotient gradient system (QGS)*, while the second stage is to compute multiple local optimal solutions in each feasible components by finding multiple stable equilibrium points in a corresponding *projected gradient system (PGS)*.

4.2 Interior Point Methods

Interior-point methods (IPMs) in mathematical programming have been the largest and most dramatic area of research in optimization since the development of the simplex method [1, 91, 121, 129, 139, 145, 174, 175]. In particular, these methods provide an attractive alternative to active set strategies in handling problems with large numbers of inequality constraints [168]. Over the past decades, there has also been a better understanding of the convergence properties of IPMs [167] and efficient algorithms have been developed with desirable global and local convergence properties [164]. IPMs have been considered as one class of the most popular methods for solving large-scale nonlinear optimization problems [19, 59, 119, 133, 169].

An IPM consists of three crucial elements: (i) the barrier method to handle inequality constraints; (ii) the Lagrangian method to handle equality constraints; and (iii) the improved Newton method to solve the set of nonlinear equations

which originates from the KKT optimality conditions. In order to solve the optimization problem (4.1), an IPM firstly applies the Fiacco-McCormick barrier method and adds slack variables to transform the optimization problem (4.1) into the following equality-constrained optimization problem:

$$\begin{array}{ll} \min_x & f(x) - \mu \sum_{j=1}^{s} \ln u_i \\ s.t. & h_i(x) = 0 \\ & g_j(x) + u_j = 0 \\ & u_j > 0 \end{array} . \tag{4.12}$$

Then the following augmented Lagrangian function can be defined:

$$L_g = f(x) - y^T h(x) - w^T [g(x) + u] - \mu \sum_{i=1}^{r} \ln u_i, \tag{4.13}$$

where, y and w are Lagrangian multipliers for equality and inequality constraints, respectively. The KKT first-order necessary conditions for the Lagrangian function L_g are given as follows:

$$\begin{array}{lll} L_x = \nabla_x f(x) - \nabla_x h(x) y - \nabla_x g(x) w & = 0 \\ L_y = h(x) & = 0 \\ L_w = g(x) + u & = 0 \\ L_u = UWE + \mu E & = 0 \end{array} , \tag{4.14}$$

where, $U = diag(u_1, u_2, \cdots, u_r)$, $W = diag(w_1, w_2, \cdots, w_r)$ and $E = [1, 1, \cdots, 1]^T$. Applying the Newton method to solve the above nonlinear equations, we can get the following two decomposed linear equations:

$$\begin{bmatrix} H & \nabla_x h(x) \\ \nabla_x^T h(x) & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} L'_x \\ -L_y \end{bmatrix} \tag{4.15}$$

and

$$\begin{bmatrix} U & W \\ 0 & I \end{bmatrix} \begin{bmatrix} \Delta w \\ \Delta u \end{bmatrix} = \begin{bmatrix} L'_u \\ -L_w - \nabla_x^T g(x) \Delta x \end{bmatrix}, \tag{4.16}$$

where

$$\begin{aligned} H &= \nabla_x^2 f(x) + \nabla_x^2 h(x) y + \nabla_x^2 g(x) U^{-1} W \nabla_x^T g(x) \\ L_x' &= L_x + \nabla_x g(x)[U^{-1}(L_u' - W L_w)] \\ L_u' &= UWE + \mu E + \Delta u \Delta w \end{aligned} \quad . \tag{4.17}$$

Observing equations (4.15)-(4.16), it is impossible to solve (4.15)-(4.16) directly because the right-hand side includes unknown high-order deviations $\Delta u \Delta w$. In order to solve this problem, the predictor-corrector interior point method was proposed, in which a predictor step and a corrector step are needed at each iteration. Because the predictor step and the corrector step share the same coefficient matrix with two different right-hand sides, only one LU factorization is needed. The predictor-corrector interior point method is composed of the following steps:

1) Initialization: Set iteration number $k = 0$, give the initial values of state variables x^0, slack variables u^0, Lagrange multipliers y^0, w^0.
2) Let $\mu = 0$, $\Delta u \Delta w = 0$, solve the linear equations (4.15) to obtain the affine direction Δx_{of}, Δy_{of}, then obtain Δu_{of}, Δw_{of} by back substitution of (4.16).
3) Compute step sizes α_{ofp} and α_{ofd}, and modify complementary gaps: GAP and GAP_{of}, update the barrier parameter:

$$\begin{aligned} \alpha_{ofp} &= \min\left\{0.9995 \min_i \left(\frac{-u_i}{\Delta u_{ofi}}, \Delta u_{ofi} < 0\right), 1\right\} \\ \alpha_{ofd} &= \min\left\{0.9995 \min_i \left(\frac{-w_i}{\Delta w_{ofi}}, \Delta w_{ofi} > 0\right), 1\right\} \\ GAP &= -u^T w \\ GAP_{of} &= -(u + \alpha_{ofp} \Delta u_{of})^T (w + \alpha_{ofd} \Delta w_{of}) \\ \mu_{of} &= \min\left\{\left(\frac{GAP_{of}}{GAP}\right)^2, 0.1\right\} \frac{GAP_{of}}{2r} \end{aligned} \quad . \tag{4.18}$$

4) Set $\mu = \mu_{of}$, $\Delta u \Delta w = \Delta u_{of} \Delta w_{of}$, and resolve the linear equations (4.14) using the same LU factorization matrix obtained in step 2 to obtain centering-corrector direction Δx, Δy. Then obtain Δu, Δw by back substitution of (4.15), and update all the variables:

$$\begin{aligned} \alpha_p &= \min\left\{0.9995 \min_i\left(\frac{-u_i}{\Delta u_i}, \Delta u_i < 0\right), 1\right\} \\ \alpha_d &= \min\left\{0.9995 \min_i\left(\frac{-w_i}{\Delta w_i}, \Delta w_i > 0\right), 1\right\} \\ x &= x + \alpha_p \Delta x \\ I &= I + \alpha_p \Delta I \\ u &= u + \alpha_p \Delta u \\ y &= y + \alpha_d \Delta y \\ z &= z + \alpha_d \Delta z \\ w &= w + \alpha_d \Delta w \end{aligned} \quad . \tag{4.19}$$

5) Compute complementary gap $GAP = -u^T w$. If GAP and maximal absolute power flow mismatch are less than the given precision, or the maximal iteration is reached, then stop; otherwise, go to step 2.

4.3 The TRUST-TECH Based IPM

Despite their nice numerical efficiency, IPMs are essentially a class of local optimization methods. In other words, from a starting point, IPMs can only compute a single local optimal solution to the optimization problem (4.1). Same as other types of local methods, there is no mechanism available in IPMs to escape from a local optimal solution and to approach another local optimal solution.

We believe that one reliable way to find the global optimal solution of the

optimization problem (4.1) is to find first all the local optimal solutions and then find, from them, the global optimal solution. This motivates us to integrate the concept of TRUST-TECH into IPMs and to develop the *TRUST-TECH based IPM (TT-IPM)* which is capable of computing all local optimal solutions to the nonlinear optimization problem (4.1).

4.3.1 The proposed method

TT-IPM finds all the local optimal solutions to the nonlinear optimization problem (4.1) using some trajectories of a particular class of nonlinear dynamical systems. TT-IPM is conceptually composed of two steps:

Step 1: Start from an arbitrary point and compute a local optimal solution to the optimization problem (4.1).

Step 2: Move away from the local optimal solution and approach another local optimal solution of the optimization problem (4.1).

The principal task in developing TT-IPM is to design a nonlinear dynamical system whose trajectories can be explored to perform Step 1 and Step 2. The central idea in designing such a nonlinear dynamical system is that all the local optimal solutions to the optimization problem (4.1) corresponds with all the *stable equilibrium points (SEPs)* of the nonlinear dynamical system. In particular, every local optimal solution to the optimization (4.1) corresponds with an SEP of the nonlinear dynamical system.

As has been shown in Section 4.2, IPM solve the optimization problem (4.1) via finding a stationary point of the penalized Lagrange function

$$L(x, y, z, s, \mu) = f(x) + y^T h(x) + z^T (g(x) + s) - \mu \sum_{i=1}^{\mathcal{I}_e} \ln s_i. \tag{4.20}$$

Specifically, a stationary point of (4.20) can be obtained by solving the KKT optimality conditions (first order necessary conditions), which can be represented as the following system of nonlinear equations:

$$\begin{array}{rcll} L_x &=& \nabla f(x) - \nabla^T h(x) y - \nabla^T g(x) z &= 0 \\ L_y &=& h(x) &= 0 \\ L_z &=& g(x) + s &= 0 \\ L_s &=& Zs - \mu e &= 0 \end{array} . \tag{4.21}$$

As $\mu \to 0$, the solution of the system of nonlinear equations (4.21) approaches a critical point $\overline{x}$ of the nonlinear optimization problem (4.1) (with Lagrange multipliers $(\overline{y}, \overline{z})$).

In order to perform Step 1 and Step 2, we build the following so-called *KKT gradient system* whose vector field is associated with critical points of the optimization problem (4.1) characterized by the gradient vector $F(w)$ of the Lagrange function (4.20):

$$\dot{w} = -\nabla^T F(w) \cdot F(w), \tag{4.22}$$

where $w = (x, y, z, s)^T$ and $F(w) = (L_x, L_y, L_z, L_s)^T$.

Based on this construction, steps 1 and 2 in the TT-IPM method can be numerically implemented via the following tasks:

Task 1: Approach an SEP of the KKT gradient system (4.22), which is a critical point of the optimization problem (4.1).

Task 2: Move from the SEP to a decomposition point (in order to escape from the critical point).

Task 3: Approach another SEP of the KKT gradient system (4.22), which is another critical point of the optimization problem (4.1), by moving along the unstable manifold of the decomposition point.

Task 1 can be implemented by following the trajectory of the KKT gradient system (4.22) starting from any initial point. Task 3 can be implemented by following the trajectory starting from an initial point, which is close to the decomposition point but outside the stability region, until it approaches another SEP of (4.22), which is another critical point of the optimization problem (4.31).

4.3.2 Characterization of the KKT gradient system

We will employ certain trajectory of the KKT gradient system (4.22) to systematically locate the set of critical points to the optimization problem (4.1) in a deterministic manner. The global dynamical behaviours of the KKT gradient system (4.22) play a central role in finding the set of critical points of the optimization problem (4.1).

Assumption 4.3.1 *For any bounded, closed interval* $[a, b]$ *and for any relatively closed subset* K *of* $(L^{-1}([a, b]) \setminus E_L)$ *in* S*, we have*

$$\inf\{\|\nabla^T F(w) \cdot F(w)\| : x \in K\} > 0, \tag{4.23}$$

where, $E_L = \{w \in R^m : \nabla^T F(w) \cdot F(w) = 0\}$ *and* $m = n + n_{\mathcal{E}} + 2n_I$.

It needs to be noticed that if each path-connected feasible component of M is compact, then Assumption 4.3.1 always holds. The constraint components of many practical optimization problems are compact, hence, Assumption 4.3.1 usually holds.

The global behaviour of nonlinear dynamical systems can be very complicated. Their trajectories can behave unbounded or/and bounded but complicated such as almost periodic trajectory, or even chaotic motions. We next show that the global behaviour of the KKT gradient system (4.22) is very simple: every trajectory converges to an equilibrium point. There is no complicated behaviours such as closed orbit (i.e., limit cycles) and chaotic motion that can exist in the KKT gradient system (4.22). Note that a nonlinear dynamical system is said to be completely stable if every trajectory of the system converges to one of its equilibrium points.

Theorem 4.3.1 (Completely Stable) *If the optimization problem (4.1) satisfies Assumption 4.3.1, then the corresponding KKT gradient system (4.22) is completely stable.*

Proof Let $V(w) = \frac{1}{2}\|F(w)\|^2$. Then

(i) For all $w \in E_L$, $\nabla V(w) = \nabla^T F(w) \cdot F(w) = 0$.

(ii) $\frac{d}{dt}V(w) = -\|\nabla^T F(w) \cdot F(w)\|^2 < 0$.

Hence, $V(w)$ satisfies the conditions required for being an energy function of the KKT gradient system (4.22). Therefore, according to [35], the KKT gradient system (4.22) is completely stable. ■

Theorem 4.3.2 (Equilibrium Points and Critical Points [104]) *If the optimization problem (4.1) satisfies Assumption 4.3.1 and all the equilibrium points of the KKT gradient system (4.22) is hyperbolic and finite in number, then each critical points of the optimization problem (4.1) is an SEP of the KKT gradient system (4.22). Conversely, if $\bar{x}$ is an SEP of the KKT gradient system (4.22), then $\bar{x}$ is an isolated local minimum of the following minimization problem*

$$\min_{w \in R^m} \quad \frac{1}{2}\|F(w)\|^2. \tag{4.24}$$

Theorem 4.3.2 provides a theoretical basis for the TT-IPM method. It asserts that each critical point of the optimization problem (4.1) can be located via the stable equilibrium point of the KKT gradient system (4.22).

Theorem 4.3.3 (Characterization of Quasi-stability Boundary [104, 33]) *Let $A_p(x_s)$ be the quasi-stability region of x_s of the KKT gradient system (4.22) and let x_i, $i = 1, 2, \cdots$, be all the decomposition points of x_s. If the system satisfies the following conditions*

i) The equilibrium points on $\partial A_p(x_s)$ are hyperbolic and finite in number;

ii) The stable and unstable manifolds of the equilibrium points on $\partial A_p(x_s)$ satisfy the transversality condition;

then the quasi-stability boundary $\partial A_p(x_s)$ is the union of the closure of the stable manifolds of x_i. In other words,

$$\partial A_p(x_s) = \bigcup_{x_i \in \partial A_p(x_s)} \overline{W^s(x_i)}. \tag{4.25}$$

We next derive a dynamical relationship between an SEP and a *dynamic decomposition point (DDP)* lying on the quasi-stability boundary of the SEP in the following theorem.

Theorem 4.3.4 (Decomposition Points and SEPs [32]) *Let $A_p(x_s)$ be the quasi-stability region of x_s of the KKT gradient system (4.22) that satisfies the following conditions*

i) The equilibrium points on quasi-stability boundaries are hyperbolic and finite in number;

ii) The stable and unstable manifolds of the equilibrium points on quasi-stability boundaries satisfy the transversality condition.

If x_d is a decomposition on the quasi-stability boundary $\partial A_p(x_s)$, then there exists another one and only one SEP, says $\bar{x}_s$, to which the unstable manifold of x_d converges. Conversely, if the set $\partial A_p(x_s) \cap \partial A_p(\bar{x}_s)$ is nonempty, then the set contains a decomposition point.

Theorem 4.3.4 reveals a relationship between SEPs and decompositions. The unstable manifold fo a dynamic decomposition point converges to two SEPs of the KKT gradient system (4.22). In the context of optimization, Theorem 4.3.3 asserts that tow neighbouring critical points are connected by the unstable manifolds of the corresponding dynamic decomposition point. Note that the two conditions stated in Theorem 4.3.3 are generic properties for general nonlinear dynamical systems.

4.3.3 Numerical methods

We now present a numerical TT-IPM method, given a local optimal solution, say x_0^s and a set of search directions, for computing tier-one local optimal solutions of x_s^0. Each tier-one local optimal solution is computed via computing the corresponding DDPs and then following the unstable manifold of each DDP and via carrying out IPMs.

TRUST-TECH Method for Computing Tier-One Local Optimal Solutions

Input: an initial local optimal solution x_s^0.

Output: the set of tier-one local optimal solutions V_s of x_s^0.

Algorithm:

Step 1: Initialize the set of decomposition points $V_d = \emptyset$ and set of tier-one local optimal solutions $V_s = \emptyset$.

Step 2: Define a set of search paths S_i for x_s^0, $i = 1, 2, \cdots, m_j$, and set $i = 1$.

Step 3: For each search path S_i starting from x_s^0, compute its corresponding local optimal solution using the following steps:

i) Apply the method for computing DDP to system (4.22) to find the corresponding DDP. If a DDP is found, proceed to the next step; otherwise, go to step vi).

ii) Letting the found DDP be denoted as x_{di}, check if it belongs to the set V_d, i.e., $x_{di} \in V_d$. If it does, go to step vi); otherwise, set $V_d = V_d \cup x_{di}$ and proceed to the next step.

iii) Set $x_{0i} = x_s^0 + (1 + \epsilon)(x_{di} - x_s^0)$, where ϵ is a small number.

iv) Starting from x_{0i}, conduct a transitional search by integrating dynamic system (4.22) to obtain the corresponding system trajectory until it reaches an interface point, say x_{fi}, at which IPMs outperforms the transitional search.

v) Starting from the interface point x_{fi} chosen in step iv), apply an IPM to find the corresponding local optimal solution with respect to x_{di}, denoted as x_s^i. And set $V_s = V_s \cup x_s^i$.

vi) Set $i = i + 1$.

vii) Repeat steps i) through vi) until $i > m_j$.

We next present a TT-IPM method for computing tier-one, tier-two, $\cdots$ local optimal solutions. Starting from an initial point, say x_0, the solution algorithm computes a complete set of local optimal solutions as well as the global optimal solution to the constrained optimization problem (4.1).

TRUST-TECH Method for Computing All Local Optimal Solutions

Input: an initial point x_0.

Output: the set of all local optimal solutions V_s to the constrained optimization problem (4.1).

Algorithm:

Step 1: Starting from x_0, apply an IPM to find a local optimal solution, say x_s^0.

Step 2: Set $j = 0$. Initialize the set of local optimal solutions $V_s = \{x_s^0\}$, the set of tier-j local optimal solutions $V_{new}^j = \{x_s^0\}$, and the set of found decomposition points $V_d = \emptyset$.

Step 3: Initialize the set of tier-$(j+1)$ local optimal solutions $V_{new}^{j+1} = \emptyset$.

Step 4: For each local optimal solution in V_{new}^j, say, x_s^j, find its all tier-one local optimal solutions.

i) Define a set of search paths S_i^j for x_s^j, $i = 1, 2, \cdots, m_j$ and set $i = 1$.

ii) For each search path S_i^j starting from x_s^j, apply an effective method to nonlinear system (4.22) to find the corresponding DDP. If a DDP is found, proceed to the next step; otherwise, go to the step viii).

iii) Letting the found DDP be denoted as $x_{d,i}^j$, check if it belongs to the set V_d, i.e., $x_{d,i}^j \in V_d$. If it does, go to step viii); otherwise, set $V_d = V_d \cup \{x_{d,i}^j\}$ and proceed to the next step.

iv) Set $x_{0,i}^j = x_s^j + (1+\epsilon)(x_{d,i}^j - x_s^j)$, where ϵ is a small number.

v) Starting from $x_{0,i}^j$, conduct a transitional search by integrating the nonlinear dynamic system (4.22) to obtain the corresponding system trajectory until it reaches an interface point. $x_{f,i}^j$, at which IPMs outperform the transitional search.

vi) Starting from the interface point $x_{f,i}^j$ chosen in step v), apply an IPM to find the corresponding local optimal solution, denoted as $x_{s,j}^i$.

vii) Check if x_{sj}^i has been found before, i.e., $x_{sj}^i \in Vs$. If it is a new local optimal solution, set $V_s = V_s \cup \{x_{sj}^i\}$, $V_{new}^{j+1} = V_{new}^{j+1} \cup \{x_{sj}^i\}$.

viii) Set $i = i + 1$.

ix) Repeat steps ii) through viii) until $i > m_j^k$.

Step 5: If the set of all newly computed local optimal solutions, V_{new}^{j+1}, is empty, continue with step 6; otherwise set $j = j + 1$ and go to step 3.

Step 6: Output the set of local optimal solutions V_s and identify the best optimal solution from the set of V_s by comparing their corresponding objective function values in and selecting the one with the smallest value. And output it as the global optimal solution.

4.4 The Reduced Projected Gradient Method

In general, the constraint set M can be very complicated with several disjoint path-connected feasible components. In other words, the constraint set M can be decomposed into several disjoint path-connected feasible components, say

$$M = \bigcup_k M_k, \tag{4.26}$$

where, $M_i \cap M_j = \emptyset, \forall i \neq j$. Each path-connected component may contain several local optimal solutions to the optimization problem (4.10). This motivates us to develop a two-phased method to solve the optimization problem (4.10), where the first phase is to compute all feasible components and the second phase is to compute all local optimal solutions in each feasible component.

4.4.1 TRUST-TECH based 2-phased method

A TRUST-TECH based method has been developed for computing multiple local optimal solutions to the optimization problem (4.1) [101, 104, 36, 31]. This

method is composed of two distinct phases: In phase I, starting from an arbitrary starting point, it systematically finds all the feasible components that constitute the constraint set M. In phase II, it computes all the local optimal solutions in each feasible component M_k found in phase I.

Phase I

Phase I of the TRUST-TECH method finds all the feasible components via exploring some trajectories of a particular nonlinear dynamical system. In order to visit each feasible component, phase I consists of two steps:

Step 1.1: Approach a (path-connected) feasible component of the optimization problem (4.10).

Step 1.2: Move away from the feasible component and approach another feasible component of the optimization problem (4.10).

We design a nonlinear dynamical system whose trajectories can be used to perform step 1.1 and step 1.2. The central idea in designing such a nonlinear dynamical system is that every path- connected feasible component corresponds with a stable equilibrium manifold of the nonlinear dynamical system. In this way, the task of finding all the feasible components of the optimization problem (4.10) is equivalent to the task of finding all the stable equilibrium manifolds of the nonlinear dynamical system. To this end, we build the following so-called *quotient gradient system (QGS)* whose vector field is associated with the constraint set of the optimization problem (4.10) characterized by the equality vector $h(x)$

$$\dot{x} = -\nabla^T h(x) \cdot h(x), \tag{4.27}$$

where $\nabla h(x)$ represents the Jacobian matrix of the vector $h(x)$.

It can be shown [103] that if a set is a feasible component of the optimization problem (4.10) then the set is a stable equilibrium manifold of the QGS (4.27). This analytical result provides a basis to search for feasible components of the optimization problem (4.10). Hence, steps 1.1 and 1.2 are numerically implemented via the following two tasks:

Task 1.1: Approach a stable equilibrium manifold of the QGS (4.27).

Task 1.2: Move away from the stable equilibrium manifold and approach another stable equilibrium manifold of the QGS (4.27).

Phase II

Phase II of the TRUST-TECH method finds all the local optimal solutions lying within each feasible component using some trajectories of a particular class of nonlinear dynamical systems. Phase II is conceptually composed of two steps:

Step 2.1: Start from a feasible point found in step 1.1, compute a local optimal solution located in the feasible component to the optimization problem (4.10).

Step 2.2: Move away from the local optimal solution and approach another local optimal solution lying in the same feasible component of the optimization problem (4.10).

A nonlinear dynamical system can be designed whose trajectories can be explored to perform step 2.1 and step 2.2. The central idea in designing such a

nonlinear dynamical system is that all the local optimal solutions to the optimization problem (4.10) correspond with all the SEPs of the nonlinear dynamical system; in particular, every local optimal solution of the optimization problem (4.10) corresponds with an SEP of the nonlinear dynamical system. To this end, the following projected gradient dynamical system is designed in [104, 36]

$$\dot{x} \;=\; -P_H(x(t))\nabla f(x(t)), \tag{4.28}$$

where, $x(0) = x_0 \in M$, and the projection matrix

$$P_H(x) \;=\; \Big(I - \nabla^T h(x)(\nabla h(x)\nabla^T h(x))^{-1}\nabla h(x)\Big) \in R^{n\times n} \tag{4.29}$$

is a positive semi-definite matrix for every $x \in M$.

Based on this construction, steps 2.1 and 2.2 can be numerically implemented via the following tasks:

Task 2.1: Approach an SEP of the projected gradient system (4.28).

Task 2.2: Move away from the SEP and approach another SEP of the projected gradient system (4.28) (in the same path-connected feasible component of the optimization problem (4.10)).

4.4.2 The proposed method

Practical optimization problems generally possess special structures that can be taken advantage of in designing effective numerical implementations. One of the most important structures is the sparsity shown in the objective Hessian matrix and the constraint Jacobian (and Hessian matrices). The existence of sparsity makes it possible to tackle large-scale optimization problems (with thousands of variables and constraints) using limited computational resources.

It needs to be noticed that the sparsity structure of the optimization problem (4.10) can be destroyed in Phase II when the dynamical system is defined as (4.28). As shown in (4.29), the projection matrix $P_H(x)$ involves a component $\tilde{H} = (\nabla h(x)\nabla^T h(x))^{-1}$. For large-scale problems, the constraint Jacobian $\nabla h(x)$ is generally sparse. However, since the inversion of a sparse matrix is generally not sparse, the existence of $\tilde{H}$ results in that the projection matrix $P_H(x)$ is no longer sparse. In other words, it could be difficult to achieve a computationally efficient implementation of Phase II for large-scale problems. This motivates us to design another dynamical system in Phase II such that the sparsity of the optimization problem (4.10) can be preserved.

It is obvious that a feasible solution to the optimization problem (4.10) is also a solution to the scalar equation

$$\tilde{h}(x) = \frac{1}{2}\|h(x)\|^2 = 0. \tag{4.30}$$

Indeed, the function $\tilde{h}(x)$ is an energy function of the projected gradient system (4.27) [104]. Using $\tilde{h}(x)$, we can now define the following optimization problem:

$$\begin{array}{ll} \min_x & f(x) \\ s.t. & \frac{1}{2}\|h(x)\|^2 = 0 \end{array}. \tag{4.31}$$

It is obvious that the optimization problem (4.31) is equivalent to the original optimization problem (4.10). Hence, the task of Phase II to find multiple local optimal solutions to the optimization problem (4.10) can be achieved by finding multiple local optimal solutions to the optimization problem (4.31).

To solve the optimization problem (4.31), the PGS in Phase II of the TRUST-TECH based method can be defined as

$$\dot{x} = -P_{\tilde{H}}(x)\nabla f(x), \tag{4.32}$$

where, $P_{\tilde{H}}(x)$ is the projection matrix defined as

$$P_{\tilde{H}}(x) = I - \frac{\nabla\tilde{h}(x)\nabla^T\tilde{h}(x)}{||\nabla\tilde{h}(x)||^2} \tag{4.33}$$

and

$$\nabla\tilde{h}(x) = \nabla^T h(x) \cdot h(x). \tag{4.34}$$

Compared with the projection matrix $P_H(x)$ defined in (4.29), there is no inversion of sparse matrix involved in $P_{\tilde{H}}(x)$. Hence, sparsity of the constraint Jacobian $\nabla h(x)$ can be preserved in Phase II. This feature of the new projection system (4.32) makes it possible to design efficient numerical implementations for solving large-scale problems using the two-phase TRUST-TECH based method.

Certain trajectories of the projected gradient dynamical system (4.32) is used to find a set of local optimal solution to the optimization problem (4.31), thus to the optimization problem (4.10). We note that $P_{\tilde{H}}(x)\nabla f(x)$ is the orthogonal projection of $\nabla f(x)$ to the tangent space T_xM, which means

$$P_{\tilde{H}}(x)\nabla f(x) \in T_xM, \ \forall x \in M. \tag{4.35}$$

Hence, every trajectory of (4.32) starting from $x_0 \in M_k$ stays in M_k. In other words, M_k is an invariant set of (4.32). This implies that the trajectory of the system (4.32) starting from a feasible point of the optimization problem (4.31) stays in the feasible component containing the point. Furthermore, it can be shown that the task of finding a local optimal solution of the optimization problem (4.31) is equivalent to the task of finding an SEP of the PGS (4.32).

Based on this construction, steps 2.1 and 2.2 in Phase II of the TRUST-TECH based method can be numerically implemented via the following tasks:

Task 3.1: Approach an SEP of the projected gradient system (4.32).

Task 3.2: Move from the SEP to a decomposition point (in order to escape from the local optimal solution).

Task 3.3: Approach another SEP of the projected gradient system (4.32) (in the same path-connected feasible component of the optimization problem (4.10)) by moving along the unstable manifold of the decomposition point.

Task 3.1 can be implemented by following trajectories in the system (4.32) starting from any initial point located in a feasible component. Task 3.3 can be implemented by following the trajectory starting from an initial point, which is close to the decomposition point but outside the stability region, until it approaches another SEP of (4.32), which is another local optimal solution to (4.31).

4.5 Numerical Results

In this section, the proposed TT-IPM and reduced projected gradient method are used to solve two constrained nonlinear programs.

4.5.1 Example 1

The first test problem is a five-dimension optimization problem:

$$\begin{aligned} \min_{x\in R^5} \quad & f(x) = (x_1-1)^2 + (x_1-x_2)^2 + (x_2-x_3)^3 + (x_3-x_4)^4 + (x_4-x_5)^4 \\ s.t. \quad & x_1 + x_2^2 + x_3^3 = 3\sqrt{2} + 2 \\ & x_2 - x_3^2 + x_4 = 2\sqrt{2} - 2 \\ & x_1 x_5 = 2 \\ & -5 \le x_i \le 5,\ i = 1, \cdots, 5 \end{aligned} \quad . \quad (4.36)$$

Table 4.1: The result of TT-IPM on the test problem 1.

ID	x					λ			$f(x)$
	x_1	x_2	x_3	x_4	x_5	λ_1	λ_2	λ_3	
1	0.728	-2.245	0.780	3.681	2.747	-17.703	-100.994	4.477	64.8719
2	-0.703	2.636	-0.096	-1.798	-2.843	-8.387	15.139	-6.496	52.9067
3	1.117	1.220	1.538	1.973	1.791	-0.064	-0.353	0.021	0.0293
4	-2.791	-3.004	0.205	3.875	-0.717	-92.256	-584.778	-138.722	606.9965
5	-1.273	2.410	1.195	-0.154	-1.571	-2.125	-1.554	-8.936	27.8730
6	4.570	-1.252	0.472	2.303	0.438	-21.270	-50.543	5.684	44.0225

There are six local optimal solutions to (4.36), which are listed in Table 4.1.

The optimization problem (4.36) is solved using TT-IPM. The solution procedure for computing all local optimal solutions to the problem (4.36) is described as follows:

1) Set the initial point $x0 = [0.0, 0.0, 0.0, 0.0, 0.0]^T$, the objective function is $f(x_0) = 1.0$. This initial point is not feasible (the equality constraints are not satisfied).

2) Using x_0 as the initial point, apply the IPM and attain a solution $x_{s0} = [4.2798, -1.3675, 0.4528, 2.4010, 0.4673]^T$. The KKT energy of this point is 31.7999 and its objective function $f(x_{s0}) = 65.0048$. In fact, this point is not an actual local optimal solution (since the KKT energy is not close to 0).

3) Using x_{s0} as the central point, search along different directions on the KKT energy surface and find exit points. Using the exit points as the initial conditions, apply the IPM to solve problem (4.36). Two new solutions

are obtained, which are: 1) $x_{s1} = [4.5696, -1.2522, 0.4718, 2.3032, 0.4377]^T$ with KKT energy being 4.7217×10^{-10} and $f(x_{s1}) = 64.8719$; 2) $x_{s2} = [0.7280, -2.2452, 0.7795, 3.6813, 2.7472]^T$ with KKT energy being 9.1276×10^{-13} and $f(x_{s2}) = 52.9067$;

4) Using x_{s1} and x_{s2} as the starting points, search along different directions on the KKT energy surface and find exit points. Using the exit points as the initial conditions, apply the IPM to solve problem (4.36). Two new solutions are obtained, which are: 1) $x_{s3} = [1.1166, 1.2204, 1.5378, 1.9728, 1.7911]^T$ with KKT energy being 4.3324^{-15} and $f(x_{s3}) = 0.0293$; 2) $x_{s4} = [-2.7909, -3.0041, 0.2054, 3.8747, -0.7166]^T$ with KKT energy being 1.2552×10^{-7} and $f(x_{s4}) = 606.9965$;

5) Using x_{s3} and x_{s4} as the central points, search along different directions on the KKT energy surface and find exit points. Using the exit points as the initial conditions, apply the IPM to solve problem (1). Two new solutions are obtained, which are: 1) $x_{s5} = [-1.2731, 2.4104, 1.1949, -0.1542, -1.5710]^T$ with KKT energy being 8.9774×10^{-14} and $f(x_{s5}) = 27.8730$; 2) $x_{s6} = [-0.7034, 2.6357, -0.0964, -1.7980, -2.8434]^T$ with KKT energy being 1.7806×10^{-15} and $f(x_{s6}) = 44.0225$.

The whole solution procedure for obtaining all local optimal solutions by TT-IPM is illustrated in Fig. 4.1, where x_{s3} is the global optimal solution to the global optimization problem (4.36).

This example well demonstrates the capability of the proposed TT-IPM method and the following advantages of incorporating TRUST-TECH with existing local methods:

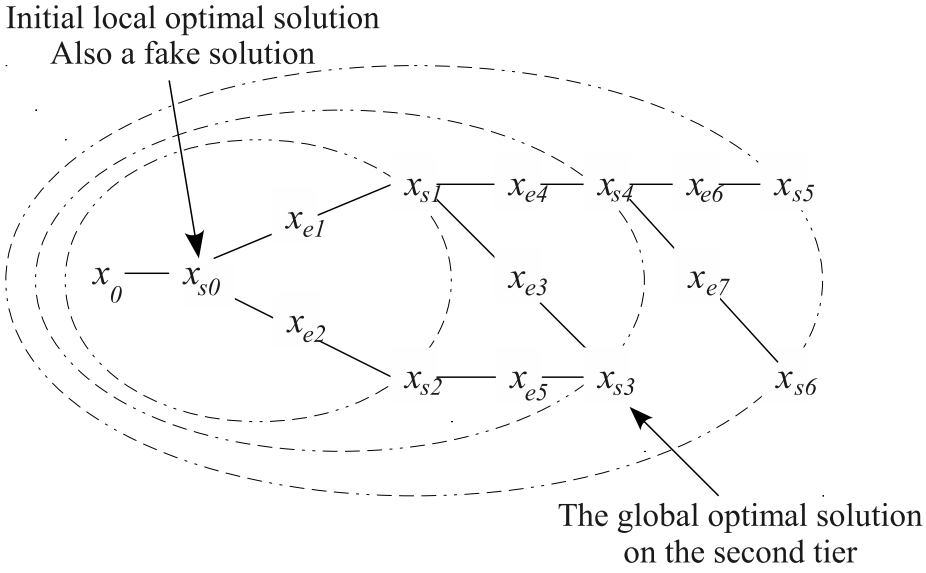


Figure 4.1: The solution procedure of TT-IPM on the optimization problem (4.36). The local method attains a fake solution starting from the initial point. TT-IPM finds all local optimal solutions, thus the global optimal solution.

1) Local optimization methods, such as IPMs, can converge to a local optimal solution and can be entrapped in the local optimal solution.

2) Local optimization methods, such as IPMs, can even converge to a point which is not a true local optimal solution.

3) The proposed TT-IPM method integrates TRUST-TECH'S capability to locate multiple stability regions in the KKT gradient system and IPM's capability to fast compute a local optimal solution, thus help local methods, such as IPMs, to escape from a local optimal solution and approach another local optimal solution.

4.5.2 Example 2

The second test problem is a two-dimensional optimization problem formulated as follows:

$$\begin{aligned} \min_{x,y} \quad & f(x,y) = -(x^2+y^2-1)^2 - ((2x^2-1)^2 + (2y^2-1)^2 - \frac{2}{3})^2 \\ s.t. \quad & 18x^2 + 18x - 42x^4 - 42x^3 + 17x^5 + 17x^6 - 3xy + 2y - 8y^2 \\ & -8y^3 + 16y^4 - 2 \leq 0 \\ & -2 \leq x, y \leq 2 \end{aligned} \tag{4.37}$$

Using the TT-IPM method

The optimization problem (4.37) is first solved using the TT-IPM method. Starting from the initial point $x_0 = (0, 0)$, TT-IPM finds eight local optimal solutions to (4.36), which are listed in Table 4.1. The TT-IPM process for solving this problem is illustrated in Fig. 4.2, where the global optimal solution, $x_{s3} = (1.272, 1.364)$, is found in the first tier local optimal solutions.

Using the reduced projected gradient method

The optimization problem (4.37) is then solved using the reduced projected gradient method. As shown in Fig. 4.3, there are two disconnected feasible components, E_1 and E_2, for this problem. To solve the problem (4.37), the inequality constraint is first converted to an equality constraint by adding a slack variable:

$$\begin{aligned} \min_{x,y,s} \quad & f(x,y) = -(x^2+y^2-1)^2 - ((2x^2-1)^2 + (2y^2-1)^2 - \frac{2}{3})^2 \\ s.t. \quad & 18x^2 + 18x - 42x^4 - 42x^3 + 17x^5 + 17x^6 - 3xy + 2y - 8y^2 \\ & -8y^3 + 16y^4 - 2 + s^2 = 0 \end{aligned} \tag{4.38}$$

Table 4.2: The result of TT-IPM on the test problem 1.

ID	x		λ					$f(x)$
	x	y	λ_1	λ_2	λ_3	λ_4	λ_5	
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-2.778
1	1.395	0.059	2.486	2.9×10^{-5}	4.9×10^{-5}	1.7×10^{-4}	5.2×10^{-5}	-76.301
2	-0.192	1.064	1.106	1.1×10^{-6}	6.5×10^{-7}	9.1×10^{-7}	2.1×10^{-6}	-3.233
3	1.272	1.364	7.555	1.5×10^{-6}	1.5×10^{-6}	6.9×10^{-6}	7.9×10^{-6}	-144.152
4	-1.476	-0.422	11.126	3.8×10^{-6}	1.3×10^{-6}	5.8×10^{-6}	8.3×10^{-6}	-123.028
5	0.707	0.707	5.9×10^{-9}	7.4×10^{-9}	7.4×10^{-9}	1.5×10^{-8}	1.5×10^{-8}	-0.444
6	-0.707	0.707	4.6×10^{-8}	7.7×10^{-8}	3.7×10^{-8}	3.7×10^{-8}	7.7×10^{-8}	-0.444
7	0.756	-0.690	0.022	1.4×10^{-7}	3.1×10^{-7}	3.2×10^{-8}	1.4×10^{-7}	-0.416

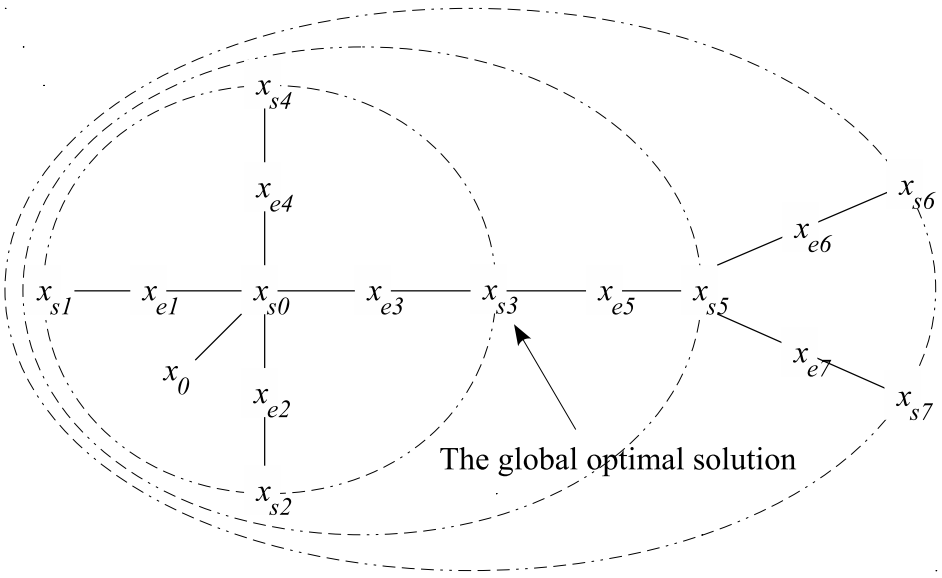


Figure 4.2: The solution procedure of TT-IPM on the optimization problem (4.37). TT-IPM finds the global optimal solution in tier-1 local optimal solutions.

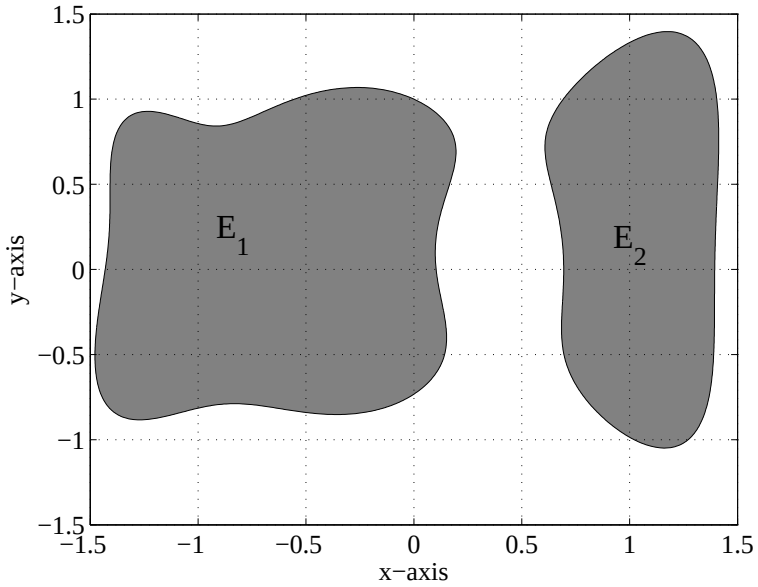


Figure 4.3: There are two disconnected feasible components, E_1 and E_2, for the optimization problem (4.37).

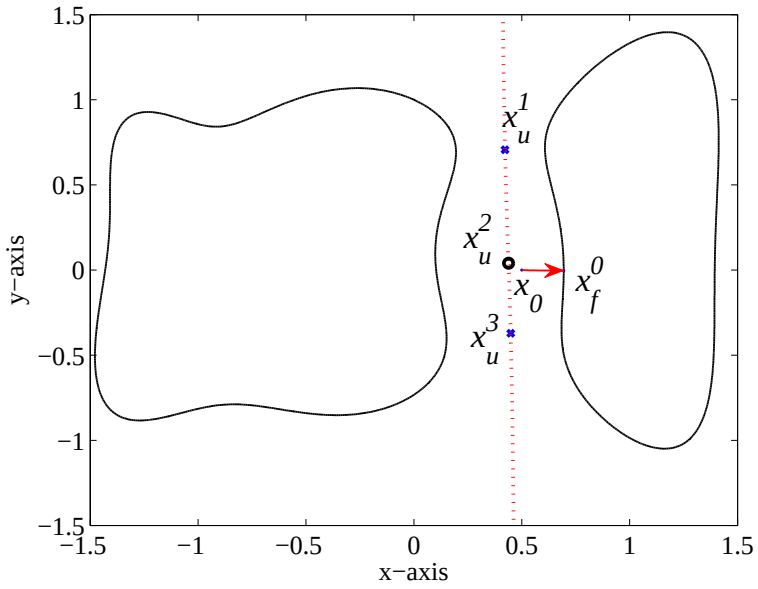


Figure 4.4: Starting from $x_0 = (0.1080, -0.0467)$, the feasible point $x_f^0 = (0.1026, 0.0078)$ in E_2 is obtained.

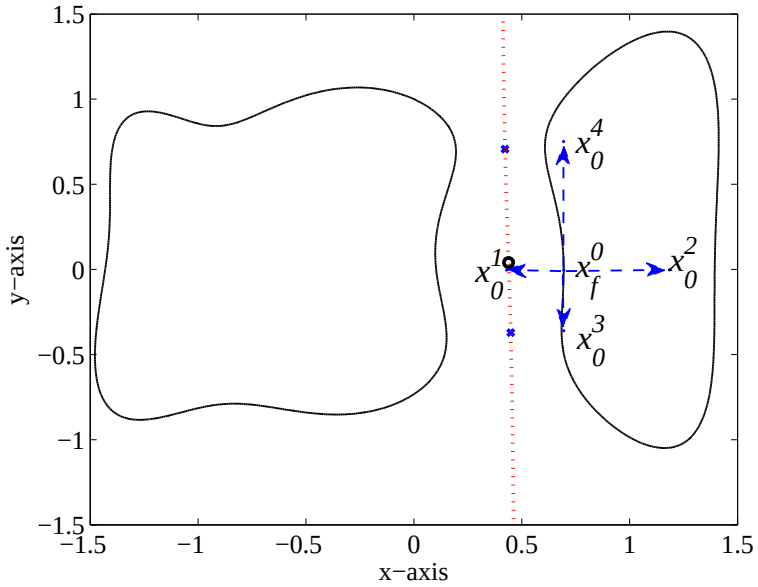


Figure 4.5: Four new initial points, x_0^1 to x_0^4, are found in the four eigendirections of the initial feasible point x_f^0.

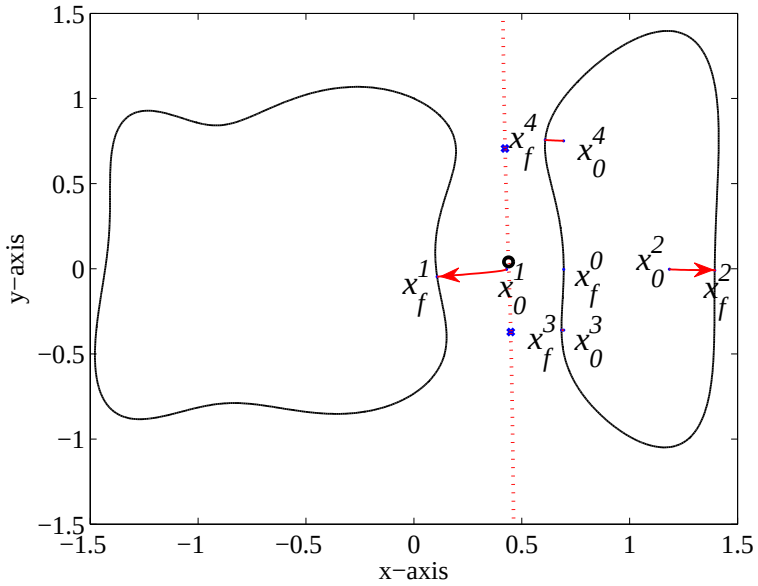


Figure 4.6: Four new feasible points, x_f^1 to x_f^4, are found by integrating the QGS system starting from x_0^1 to x_0^4, respectively. In particular, x_f^1 is in E_1, while x_f^2 to x_f^4 are in E_2

By constructing the QGS (4.27), phase I of the method first computes all feasible components through the following steps:

1) Starting from the initial point $x_0 = (0.5, 0, 0)$, the initial feasible point $x_f^0 = (0.1026, 0.0078)$ can be obtained by integrating the QGS system trajectory. This initial feasible point belongs to the feasible component E_2 (Fig. 4.4).
2) Starting from x_f^0, four search directions along the eigenvectors of the QGS Jacobian at x_f^0 are emanated from x_f^0. Four new initial points, x_0^1 to x_0^4, are located in these directions (Fig. 4.5).
3) Starting from x_0^1 to x_0^4, four new feasible points, x_f^1 to x_f^4 are reached by integrating the corresponding trajectories in the QGS (4.27). Specifically, x_f^1 belongs to the feasible component E_1, while x_f^2 to x_f^4 belong to the feasible component E_2 (Fig. 4.6).

Then, phase II of the method finds multiple local optimal solutions to the optimization problem (4.37) by finding all SEPs in the PGS (4.32). Each of the five feasible points obtained in phase I is used to initiate a multi-tier TRUST-TECH search. As a result, eleven SEPs are found in the PGS (4.32), each of which is a local optimal solution to the problem (4.37). Specifically, six local optimal solutions are found in the feasible region E_1 (Table 4.3) via the following steps:

1.1) Starting from the feasible point $x_f^1 = (0.1080, -0.0477)$, the local optimal solution $x_s^1 = (0.1026, 0.0078)$ is found by integrating the system trajectory of the PGS (4.32).
1.2) Around x_s^1, a new initial point $x_0^2 = (0.1002, 0.0093)$ is found, starting from which a tier-1 local optimal solution $x_s^2 = (0.0, 0.0)$ is found by integrating the PGS trajectory.

Table 4.3: Six local optimal solutions found in the feasible component E_1.

ID	Solution		$f(x,y)$	Tier	Direction		Initial Point	
	x	y			x	y	x	y
1	0.1026	0.0078	-2.647	0	-	-	0.1080	-0.0467
2	0.0	0.0	-2.778	1	-0.23	0.14	0.1002	0.0093
3	-0.7071	0.7071	-0.444	2	-0.42	0.49	-0.4932	0.5772
4	-0.7071	-0.7071	-0.444	2	-0.30	-0.23	-0.6211	-0.4720
5	-1.4757	-0.4220	-122.981	2	-0.32	0.06	-0.8125	0.1418
6	-0.1925	1.0641	-3.233	2	0.11	0.49	0.1794	0.7941

1.3) Around x_s^2, four new initial points, $x_0^3 = (-0.4932, 0.5772)$, $x_0^4 = (-0.6211, -0.4720)$, $x_0^5 = (-0.8125, 0.1418)$, $x_0^6 = (0.1794, 0.7941)$, are found. Starting from these initial points, four tier-2 local optimal solutions, $x_s^3 = (-0.7071, 0.7071)$, $x_s^4 = (-0.7071, -0.7071)$, $x_s^5 = (-1.4757, -0.4220)$ and $x_s^6 = (-0.1925, 1.0641)$, respectively, are located by integrating the PGS trajectories.

The remained five local optimal solutions are found in the feasible component E_2 (Table 4.4) via the following steps:

2.1) Starting from the feasible point $x_f^0 = (0.6944, -0.0032)$, $x_f^2 = (1.3942, -0.0087)$ and $x_f^4 = (0.6088, 0.7563)$, three local optimal solutions, $x_s^7 = (0.6944, 0.0005)$, $x_s^8 = (1.3950, 0.0595)$ and $x_s^9 = (0.6087, 0.6769)$, are found by integrating system trajectories of the PGS (4.32). The trajectory starting from the feasible point $x_f^3 = (0.6843, -0.3603)$ also converges to x_s^7.

2.2) Around x_s^7, a new initial point $x_e^{10} = (0.4791, 0.6760)$ is found, starting from

Table 4.4: Five local optimal solutions found in the feasible component E_2.

ID	Solution		$f(x,y)$	Tier	Direction		Initial Point	
	x	y			x	y	x	y
7	0.6944	0.0005	-0.380	0	-	-	0.6944	-0.0032
8	1.3950	0.0595	-76.300	0	-	-	1.3942	-0.0087
9	0.6087	0.6769	-0.381	0	-	-	0.6088	0.7563
10	0.7070	0.7071	-0.444	1	0.24	0.42	0.4791	0.6760
11	1.2756	1.3615	-144.121	1	0.03	0.99	1.4114	0.6405

which a tier-1 local optimal solution $x_s^{10} = (0.7070, 0.7071)$ is found by integrating the PGS trajectory.

2.3) Around x_s^8, a new initial point $x_e^{11} = (1.4114, 0.6405)$ is found, starting from which a tier-1 local optimal solution $x_s^{11} = (1.2756, 1.3615)$ is found by integrating the PGS trajectory.

The global optimal solution to the optimization problem (4.37) is found to be the tier-1 local optimal solution x_s^{11} in the feasible component E_2.

4.6 Summary

We have developed two TRUST-TECH based methods, including TT-IPM and the reduced projected gradient method, for computing multiple local optimal solutions to constrained nonlinear optimization problems. In the first method, a TRUST-TECH search is directly carried out on a KKT gradient system to search for all stable equilibrium points, each of which corresponds to a critical point of

the optimization problem. The second method consists of two distinct phases: phase I, starting from an arbitrary point, systematically finds all feasible components which satisfy nonlinear equality and inequality constraints; phase II then computes all the local optimal solutions in each feasible components found in phase I. Numerical methods have been developed to implement these methods.

From a computational viewpoint, it may well be due to the lack of computational methods for computing the complete set of decomposition points that the TRUST-TECH based methods can only compute multiple local optimal solutions (in each feasible components) for general constrained optimization problems. In addition, due to the lack of computation methods for computing the complete unstable manifolds, the TRUST-TECH-based methods can only compute multiple feasible components, instead of all of the feasible components for general constrained optimizations.

TRUST-TECH based methods are valuable and can be of significant importance in the following sense: since a sophisticated local-type method, such as IPMs, can only provide local optimal solutions in a restricted local region while a global-type method can only provide sparse approximated solutions within reasonable computational efforts, the TRUST-TECH based methods represent an enabling technology for:

- enhancing the functionality of local-type methods to effectively find optimal solutions in a broader region, as the proposed TT-IPM method;
- enhancing the functionality of global-type methods to effectively find near global optimal solutions;
- providing more flexibility in the integrating local-type and global-type methods to effectively find global optimal solutions.

CHAPTER 5

TRUST-TECH BASED METHODOLOGY FOR SOLVING MIXED INTEGER NONLINEAR PROGRAMMING

5.1 Introduction

Many applications, such as power system planning [160, 186, 189], unit commitment problems [111, 132, 152], placement of synchronized measurements [25] and capacitor placement and control [8, 39, 115, 147], fall under the category of constrained nonlinear optimization problems (NLP). On the other hand, because of the discrete controls (shunt compensation, transformer taps, etc) involved in power systems, these optimization problems are essentially mixed-integer nonlinear programs.

Mixed integer nonlinear programming (MINLP) provides a fairly general and powerful framework to model optimization problems involving both discrete and continuous variables. In this framework, the nonlinear expression better approximates real-world phenomena while its discrete variables offer great flexibility to represent an indivisible quantity or applications involving decision making. In the literature, numerous studies of MINLP problems have been reported in areas such as engineering [10, 58], biological science and economics. For instance, determination of the molecular structure of a substance based on the energy calculation is formulated as an MINLP problem [148, 161, 162]. The MINLP methodology was applied to portfolio selection in financial engineering [100, 120] and to resource allocation and scheduling problems [17, 45]. Over the past few decades, there has been a pronounced increase in the study of MINLP problems.

Although MINLP provides great flexibility and accuracy, the task of solving such problems is usually challenging. The diverse nonlinear behaviour in MINLP problems creates a great challenge not only in algorithmic developments but also in numerical implementations. Developing efficient and robust methods and tools for solving MINLP problems continues to attract a lot of research and development efforts. From a theoretical viewpoint, it is well known that the major difficulty in solving general continuous nonlinear programming problems arises from the non-convexity of objective functions and/or nonlinear constraints. Without the aid of other supplemental techniques, the performance of stand-alone, deterministic, local-type algorithms is greatly restricted by such non-convexity. The presence of integral variables further deteriorates the situation since it leads to an explosive increase in the number of solutions.

A large variety of algorithms have been proposed for solving MINLP problems. Many of these proposed algorithms basically integrate and extend schemes designed for handling non-convexity arising from nonlinear expressions and the presence of integral variables. Algorithms of this type reported in the literature include *Branch and Bound* [12, 74], *Generalized Benders Decomposition* [65], *Outer Approximation* [52], *Extended Cutting Plane Method* [172] and so forth. Details of these proposed algorithms may be found in [72].

Although schemes such as the branching or cutting plane method might alleviate adverse effects due to the presence of discrete variables, these algorithms are still limited by the non-convex nature arising from the inherent nonlinear expression. For instance, the Branch and Bound algorithm creates a sequence of sub-problems from the original MINLP problem and seeks to identify optimal solutions by solving some sub-problems. Its overall efficiency depends

heavily on whether good solutions could be identified at an early stage to help its pruning scheme. To identify potential solutions, it will be indispensable to solve some non-convex sub-problems (even without integral variables).

The TRUST-TECH methodology has been developed to solve non-convex global optimization problems [32, 36, 104] and it has been successfully applied to various applications, such as learning finite mixture models [143], constructing neural network ensembles [171], and solving optimal power flow problems [38]. One distinguishing feature of the TRUST-TECH methodology comes from its ability to systematically and efficiently locate multiple local optimal solutions in a tier-by-tier manner. With the aid of topological information from the problem structure, the issue of non-convexity can be successfully overcome. We have demonstrated its efficiency in general continuous nonlinear programming problems. In this chapter we present a TRUST-TECH based methodology to conquer the non-convexity problem both from nonlinear expression and the presence of integral variables. However, we point out that the TRUST-TECH methodology presented for solving MINLP problems is by no means the only possible extension of the TRUST-TECH methodology. There are several possible ways that the TRUST-TECH methodology can significantly contribute to solving MINLP problems. For instance, the TRUST-TECH methodology can obviously be incorporated with Branch and Bound type algorithms to enhance their effectiveness.

5.2 Mixed Integer Programming

Before we present the TRUST-TECH based methodology for solving MINLP problems, a brief review of the MINLP formulation and existing methods is first given in this section.

MINLP problems can be formulated as follows:

$$\begin{aligned} \min_{x,y} \quad & f(x,y) \\ s.t. \quad & g_j(x,y) \le 0 \text{ for } j \in J = \{1,2,\cdots,j_1\} \\ & h_i(x,y) = 0 \text{ for } i \in I = \{1,2,\cdots,i_1\} \\ & x^l \le x \le x^u,\ y^l \le y \le y^u \\ & x^l, x, x^u \in R^{n\times 1},\ y^l, y, y^u \in Z^{m\times 1} \end{aligned} \tag{5.1}$$

where J: the index set for inequality constraints and I: the index set for equality constraints. In this formulation x is the set of continuous variables that appears either linearly or/and nonlinearly in the formulation, and y represents the set of integer variables in the formulation. The objective function $f(x,y)$, inequality constraints $g_j(x,y)$, and equality constraints $h_i(x,y)$, for the time being, are all continuously differentiable functions of both the continuous variable x_k and the discrete variable y_k, where x_k denotes the k-th element of x and y_k denotes the k-th element of y. If variable y_k can only take value 0 or 1, we call the problem a mixed binary nonlinear programming problem. Otherwise, we will call the problem a mixed integer nonlinear programming problem.

The above general formulation (5.1) contains several aspects that are difficult to solve. Features such as nonlinearity in the discrete variables and non-separability of continuous and discrete variables are included. The Generalized Benders Decomposition [65] can be implemented to solve the above mixed-

integer nonlinear formulation. In addition, two other algorithms have been proven effective against this problem: the Outer Approximation algorithm [52] and the Equality Relaxation algorithm. These three algorithms make use of projection, outer approximation and relaxation. The basic idea behind all three algorithms is to solve an alternating finite sequence of nonlinear programming (NLP) subproblems, which provide upper bounds for the optimal solution, and mixed-integer linear programming (MILP) master problems, which provide lower bounds for the optimal solution.

5.3 The TRUST-TECH Based Methodology for Solving MINLPs

Compared to solving the continuous optimization problem, the major difficulty introduced in solving MINLP problems rests in how to properly handle the integral variables. However, the TRUST-TECH methodology has been developed principally for continuous optimization problems. Hence, the MINLP problem needs first to be converted to a continuous one before TRUST-TECH is applied.

On the other hand, as the dimension of the MINLP problem increases, the number of local optimal solutions usually increases as well. As a result, the number of tiers for TRUST-TECH to search local optimal solutions in the constructed continuous problem needs to be increased as well. However, if the number of tiers we define as the neighbourhood is too large, the entire searching process might be time consuming or unmanageable. This issue needs to be addressed in the proposed TRUST-TECH based methodology. Another issue that needs to be addressed is the development of a method to eliminate the issue of non-convexity arising from the presence of integral variables. Address-

ing these issues may give our TRUST-TECH based framework advantages for solving continuous constrained problems or easy integration with existing efficient solvers. We hence propose a multi-stage two-phase TRUST-TECH-based methodology for solving MINLP problems, which consists of the following basic steps:

Step 1. Convert the constrained MINLP problem (5.1) into a continuous problem where integral variables are relaxed into continuous ones;

Step 2. Apply the TRUST-TECH based methods to search for multiple solutions to the continuous problem;

Step 3. Apply a two-stage TRUST-TECH search to find a set of local optimal solutions to the original MINLP problem (5.1). Specifically, for each local optimal solution, a reduced constrained NLP is constructed by determining and fixing the values of integral variables of the MINLP problem. The TRUST-TECH based methods is then used to compute a set of local optimal solutions to these reduced NLP problems, from which the optimal solution of the original MINLP problem (5.1) is determined.

This multi-stage procedure is designed to properly manage the required computational efforts. At each stage, the search procedure of the TRUST-TECH based methodology is to explore just a small number of tiers around the starting local optimal solution in the search space which we define as the neighbourhood. Upon completion of the one-stage search, we determine whether the obtained local optimal solutions achieve a certain solution quality. If there are no satisfactory local optimal solutions found and sufficient computational time is still available, another stage of the search procedure will continue.

In the meantime, the purpose of a two-phase search procedure within each stage is to eliminate the complicating procedure of determining the optimal values for the integral variables. During the first phase, several local optimal solutions among all the local optimal solutions found are evaluated and used to determine the values of the integral variables. Each valid solution of the integral variables will lead to a corresponding reduced nonlinear programming problem (with fixed values of the integral variables).

In the remaining parts of this section, each step in the TRUST-TECH based methodology to realize this conceptual algorithm will be described in detail.

5.3.1 Problem relaxation

Our proposed TRUST-TECH-based methodology for solving the MINLP problem (5.1) first relaxes the original problem by treating all the discrete variables y_k as continuous variables. The corresponding relaxed problem is then described below:

$$\begin{aligned} \min \quad & f(x,y) \\ s.t. \quad & g_j(x,y) \le 0, \text{ for } j \in J = \{1,2,\cdots,j_1\} \\ & h_i(x,y) = 0, \text{ for } i \in I = \{1,2,\cdots,i_1\} \\ & x^l \le x \le x^u,\ y^l \le y \le y^u \\ & x^l, x, x^u \in R^{n\times 1},\ y^l, y^u \in Z^{m\times 1}.\ y \in R^{m\times 1} \end{aligned} \quad . \tag{5.2}$$

This continuous constraint problem (5.2) is then handled by the TRUST-TECH based methods, such as the TRUST-TECH based interior point method (TT-IPM) that is developed in Chapter 4. A set of local optimal solutions to the optimization problem (5.2) can be obtained after applying TT-IPM. These solu-

tions are potential local optimal solutions to the MINLP (5.1), and are denoted as (x^*, y^*).

5.3.2 Sensitivity analysis

For each potential local optimal solution (x^*, y^*), sensitivity analysis is performed to determine the value of discrete variables. Sensitivity can be evaluated as [22]

$$\begin{aligned} S_y^f &= \frac{\partial f}{\partial y} - \left(\frac{\partial h}{\partial y}\right)^T \left[\left(\frac{\partial h}{\partial x}\right)^T\right]^{-1} \frac{\partial f}{\partial x} \\ S_y^g &= \frac{\partial g}{\partial y} - \frac{\partial g}{\partial x}\left(\frac{\partial h}{\partial x}\right)^{-1} \frac{\partial h}{\partial y} \end{aligned}, \tag{5.3}$$

where, S_y^f and S_y^g represent the sensitivity of the objective function and inequality constraints with respect to the discrete variable changes, respectively.

Using the sensitivity computed with (5.3), we can compute a linear estimation of the change in the objective and the inequality constraints when moving the discrete variable y_i from its current value y_i^k to its nearest upper or lower value by

$$\begin{aligned} \Delta f_i^+ &= S_{y_i}^f (y_i^{k+1} - y_i^k) \\ \Delta f_i^- &= S_{y_i}^f (y_i^{k-1} - y_i^k) \\ \Delta g_{ij}^+ &= S_{y_i}^{g_j} (y_i^{k+1} - y_i^k), \ \forall j \in J \\ \Delta g_{ij}^- &= S_{y_i}^{g_j} (y_i^{k-1} - y_i^k), \ \forall j \in J \end{aligned}, \tag{5.4}$$

where, y_i^{k+1} and y_i^{k-1} represent the nearest upper and lower discrete values of y_i^k, respectively; f_i^+ and f_i^- are the estimated change in the objective, g_{ij}^+ and g_{ij}^- are the estimated change in the j-th inequality constraint.

Based on the linear estimations (5.4), we can then define the following two merit functions:

$$\begin{aligned} \eta_i^+ &= w_f \Delta f_i^+ + \sum_{j=1}^{J} w_g \max[0, g_j(\tilde{x}, \tilde{y}) + \Delta g_{ij}^+] \\ \eta_i^- &= w_f \Delta f_i^- + \sum_{j=1}^{J} w_g \max[0, g_j(\tilde{x}, \tilde{y}) + \Delta g_{ij}^-] \end{aligned}, \tag{5.5}$$

where, $w_f > 0$ and $w_g > 0$ are weights on the objective and constraints, respectively. These merit functions are designed to evaluate the influence caused by the movement of y_i to its nearest upper and lower discrete values by combining the variation of the objective and the variation of the amount of constraint violations. It can be observed from (5.5) that only violated inequality constraints (after altering the discrete variable) contribute to the merit functions. Obviously, the lower the value of the merit functions, the better the effect of moving a discrete variable.

5.3.3 Problem reduction

Given a potential local optimal solution (x^*, y^*) with values for discrete variables have been determined by sensitivity analysis, the corresponding reduced NLP problem can be represented as follows:

$$\begin{aligned} \min_x \quad & f(x, y^*) \\ s.t. \quad & g_j(x, y^*) \leq 0 \\ & h_i(x, y^*) = 0 \\ & x^l \leq x \leq x^u \\ & x^l, x, x^u \in R^{n \times 1} \end{aligned} \tag{5.6}$$

The reduced NLP problem (5.6) can be solved again by the TRUST-TECH

based methods, such as the TT-IPM method developed in Chapter 4 for constrained nonlinear programming problems. Using TT-IPM, we can obtain multiple local optimal solutions to the problem (5.6). Each of the computed local optimal solution to the optimization problem is a local optimal solution to the original MINLP problem (5.1).

5.3.4 The proposed TRUST-TECH-based method

At this point, we can outline three major steps for our search procedure within one stage as follows:

Step 1. Locate multiple local optimal solutions of the relaxed MINLP problem (5.1) using the TT-IPM method.

Step 2. From each local optimal solution found, determine the value for each integral variable of the given MINLP problem (5.1).

Step 3. Determine the optimal values for continuous variables of the corresponding reduced problem (5.6) of the original MINLP problem (5.1) where the optimal values of integral variables have been determined in Step 2.

We describe the basic steps of our proposed search procedure using the flow chart shown in Figure 5.1. The proposed TRUST-TECH-based methodology is presented as follows.

TRUST-TECH-based methodology for MINLP

Stage 1:

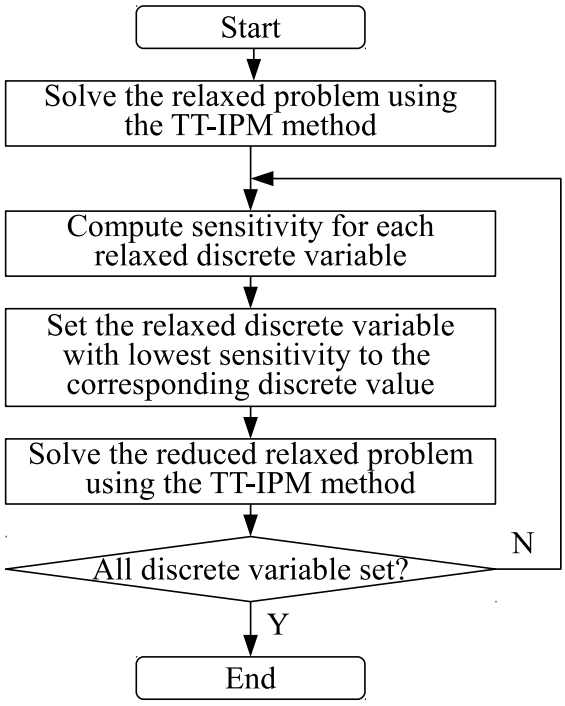


Figure 5.1: A flow chart of the TRUST-TECH-based methodology for solving MINLP problems.

Step 1.1 Initialization: Choose an initial condition for the TT-IPM method.

Step 1.2 Apply the TT-IPM method to locate multiple local optimal solutions starting from the chosen initial condition.

Step 1.3 Determine the values of the integral variables of the MINLP problem (5.1) from the local optimal solutions found in Step 1.2.

Step 1.4 Derive a reduced NLP problem (5.6) with values of the integral variables fixed at those obtained in Step 1.3.

Step 1.5 Apply the TRUST-TECH-based method to determine a set of local optimal solutions to the reduced NLP problem (5.6).

Step 1.6 Determine the optimal solution to the MINLP problem (5.1) based on the set of local optimal solutions obtained in Step 1.5.

Stage 2:

Step 2.1 Initialization: Use the best local optimal solution identified in Stage 1 as new initial condition for the TT-IPM method.

Step 2.2 Apply the TT-IPM method to locate multiple local optimal solutions starting from the chosen initial condition.

Step 2.3 Determine the values of the integral variables of the MINLP problem (5.1) from the local optimal solutions found in Step 2.2.

Step 2.4 Derive a reduced NLP problem (5.6) with values of the integral variables fixed at those obtained in Step 2.3.

Step 2.5 Apply the TRUST-TECH-based method to determine a set of local optimal solutions to the reduced NLP problem (5.6).

Step 2.6 Determine the optimal solution to the MINLP problem (5.1) based on the set of local optimal solutions obtained in Step 2.5.

The first stage applies the TRUST-TECH-based method for a breadth-first search (i.e. locate local optimal solutions lying within three to five tiers from the initial local optimal solutions). This stage identifies the top few local optimal solutions from the three-tier to five-tier local optimal solutions and uses the directions from the first local optimal solution to these top local optimal solutions as the search directions to be used in Stage 2. The second stage applies the TRUST-TECH-based method for a depth-first search (i.e. locate multiple local optimal solutions starting from the initial local optimal solution and move along the search directions determined in the first stage).

As shown in Figure 5.1, values of the discrete variables are determined one by one in each stage. At each local optimal solutions found by the TT-IPM method, merit function values for those relaxed discrete variables are computed and compared. The discrete variable with the lowest merit function value will be adjusted to the corresponding discrete value and be fixed. The best solution (with a new fixed discrete variable) is used to construct the reduced problem (with dimension decreased by one). This process is iterated until all discrete variables have been fixed.

Generally, the number of local optimal solution increases with the dimension of the optimization problem under study. If the initial condition given to the TRUST-TECH-based method is far away from high-quality local optimal solutions, the search procedure needs to go through many tiers to obtain these high-quality local optimal solutions. For large-scale optimization problems, this search procedure can be time consuming or even unmanageable. A naive approach to conquer such difficulty is to reduce the number of search directions. That is, from a located local optimal solution, we only move along a few directions to reach other nearby local optimal solutions. This approach, however, can be effective but it does not make use of any information obtained during the search process.

The issue is then how to determine promising search directions on the basis of all identified local optimal solutions. Since our TRUST-TECH method is applied to search a promising region containing high-quality local optimal solutions, we conjecture that nearby local optimal solutions have similar behaviour in the sense that they are uniform in terms of their objective values or feasibility of the original MINLP problem. With such a conjecture, high-quality local

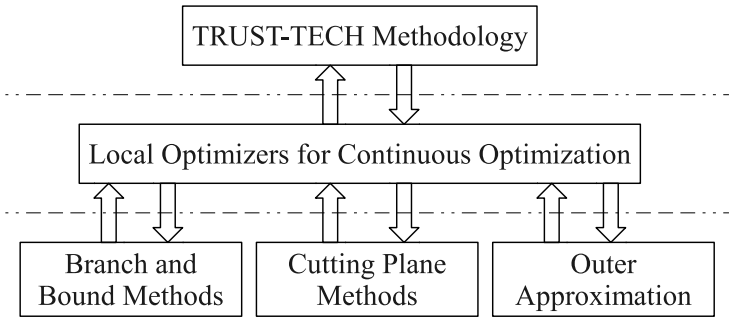


Figure 5.2: Architecture of the three-stage TRUST-TECH-based methods for solving mixed integer nonlinear programming problems.

optimal solutions will cluster together and locate themselves in certain search directions starting from one initial condition. Therefore, if a high-quality local optimal solution is identified in a certain search direction, the search process should continue towards that direction to look for other better solutions. We design a multi-stage procedure of our proposed method to realize such an idea.

During the first stage, the search procedure is focused on exploring a neighbourhood of the starting local optimal solution up to certain tiers. The number of tiers should be sufficiently small such that the search process can be completed within a reasonable amount of computational effort. In our simulations, the number of tiers is chosen to be between three and five. We use the first stage as an indication to either increase or decrease the number of tiers for the following stages depending on the computational resources allocated to each stage. Upon the completion of the first stage, a good identified local optimal solution (after rounding integral variables to their nearest values) is used as a new initial condition for the next stage. The process is repeated until a satisfactory local optimal solution is located.

There are several other options available that our TRUST-TECH methodology can significantly contribute to solving MINLP problems. For instance, the

TRUST-TECH methodology obviously can be incorporated with Branch and Bound type algorithm to enhance its pruning scheme. Figure 5.2 illustrates a framework for such incorporation of TRUST-TECH with existing algorithms.

5.4 Numerical Results

In this section, we illustrate the proposed method on three testing problems which appear in *Handbook of Test Problems in Local and Global Optimization* [57]. The weights in computing the merit values (5.5) are $w_f = 1.0$ and $w_g = 2.0$.

5.4.1 Testing problem 1

The first problem is the following seven-dimensional problem with three continuous and four binary variables:

$$
\begin{aligned}
\min \quad & (y_1-1)^2+(y_2-1)^2+(y_3-1)^2+(x_1-1)^2+(x_2-2)^2+(x_3-3)^2-\ln(y_4+1) \\
s.t. \quad & x_1+x_2+x_3+y_1+y_2+y_3 \le 5 \\
& x_1^2+x_2^2+x_3^2+y_3^2 \le 5.5 \\
& y_1+x_1 \le 1.2 \\
& y_2+x_2 \le 1.8 \\
& y_3+x_3 \le 2.5 \\
& y_4+x_1 \le 1.2 \\
& x_2^2+y_2^2 \le 1.64 \\
& x_3^2+y_3^2 \le 4.25 \\
& x_3^2+y_2^2 \le 4.64 \\
& x_1, x_2, x_3 \ge 0,\ y_i \in \{0,1\} \text{ for } i = 1, \cdots, 4.
\end{aligned}
$$

The initial point for solving this problem is $x_0 = (0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5)$. The proposed method solves this problem via the following steps:

1) Starting from x_0, TT-IPM finds a local optimal solution x_s^1=(0.2596, 1.1520, 1.9744, 0.5994, 0.5593, 0.4553, 0.9404) with objective $f(x,y) = 2.3077$. Around x_s^1, there is no higher tier local optimal solution found. The merit values computed at x_s^1 are η_1^+=(0.5994, 2.3107, 3.1396, 0.0885) and η_1^-=(0.4802, 0.4930, 0.4960, 0.4846). Hence, the discrete variable y_4 is set to 1 and x_0^1=(0.2596, 1.1520, 1.9744, 0.5994, 0.5593, 0.4553, 1).

2) Starting from x_0^1, TT-IPM finds the local optimal solution x_s^2=(0.2000, 1.1452, 1.9811, 0.6275, 0.5731, 0.4731, 1) with objective $f(x,y) = 2.3143$. Around x_s^2, there is no higher tier local optimal solution found. The merit values computed at x_s^2 are η_2^+=(0.4675, 2.3631, 3.2518) and η_2^-=(0.4675, 0.4893, 0.4986). Hence, $y_1 = 1$ and x_0^2=(0.2000, 1.1452, 1.9811, 1,0.5731, 0.4731,1).

3) Starting from x_0^2, TT-IPM finds the local optimal solution x_s^3=(0.2000, 1.1526, 2.0113, 1, 0.3423, 0.2939, 1) with objective $f(x,y) = 2.5738$. Around x_s^3, there is no higher tier local optimal solution found. The merit values computed at x_s^3 are $\eta_3^+ = (1.6670, 2.8608)$ and $\eta_3^- = (0.4502, 0.4150)$. Hence, $y_3 = 0$ and x_0^3=(0.2000, 1.1526, 2.0113, 1, 0.3423, 0, 1).

4) Starting from x_0^3, TT-IPM finds the local optimal solution x_s^4=(0.2000, 1.1138, 2.0542, 1, 0.6321, 0, 1) with objective $f(x,y) = 2.7623$. Around x_s^4, there is no higher tier local optimal solution found. The merit values computed at x_s^4 are $\eta_4^+ = 2.9112$ and $\eta_4^- = 0.4651$. Hence, $y_2 = 0$ and x_0^4=(0.2000, 1.1138, 2.0542, 1, 0, 0, 1).

5) Starting from x_0^4, TT-IPM finds the local optimal solution x_s^5=(0.2000,

1.2806, 1.9545, 1, 0, 0, 1), which is the global optimal solution to the optimization problem (5.7) and $f(x^{opt}, y^{opt}) = 3.5575$.

5.4.2 Testing problem 2

The second problem is the following five-dimensional MINLP problem with two continuous and three binary variables:

$$\begin{aligned} \min \quad & 2x_1 + 3x_2 + 1.5y_1 + 2y_2 - 0.5y_3 \\ s.t. \quad & x_1^2 + y_1 = 1.25 \\ & x_2^{1.5} + 1.5y_2 = 3 \\ & x_1 + y_1 \leq 1.6 \\ & 1.333x_2 + y_2 \leq 3 \\ & -y_1 - y_2 + y_3 \leq 0 \\ & x_1, x_2 \geq 0,\ y_i \in \{0, 1\} \text{ for } i = 1, 2, 3. \end{aligned} \tag{5.7}$$

The initial point for solving the problem (5.7) is $x_0 = (0.5, 0.5, 0.5, 0.5, 0.5)$. The proposed method happens to solve this problem in one step. Starting from x_0, TT-IPM finds a local optimal solution $x_s^1 = (1.1180, 1.3104, 0.0000, 1.0000, 1.0000)$ with objective $f(x, y) = 7.6672$. Around x_s^1, there is no higher tier local optimal solution found.

It can be observed that all discrete variables in x_s^1 are already very close to their closest discrete values (with difference smaller than 10^{-6}). Hence, the discrete variables are set to the corresponding discrete values and we obtain the global optimal solution to the optimization problem (5.7), which is $(x^{opt}, y^{opt}) = (1.1180, 1.3104, 0, 1, 1)$ and $f(x^{opt}, y^{opt}) = 7.6672$.

5.4.3 Testing problem 3

The third test problem is an eleven-dimensional MINLP problem with three continuous and eight binary variables defined as follows:

$$
\begin{aligned}
\min \quad & -x_1 x_2 x_3 \\
s.t. \quad & x_1 + 0.1^{y_1} 0.2^{y_2} 0.15^{y_3} = 1 \\
& x_2 + 0.05^{y_4} 0.2^{y_5} 0.15^{y_6} = 1 \\
& x_3 + 0.02^{y_7} 0.06^{y_8} = 1 \\
& -y_1 - y_2 - y_3 \leq -1 \\
& -y_4 - y_5 - y_6 \leq -1 \\
& -y_7 - y_8 \leq -1 \\
& 3y_1 + y_2 + 2y_3 + 3y_4 + 2y_5 + y_6 + 3y_7 + 3y_8 \leq 10 \\
& 0 \leq x_1, x_2, x_3 \leq 1, \; y_i \in \{0, 1\} \text{ for } i = 1, \cdots, 8.
\end{aligned}
\tag{5.8}
$$

The initial point for solving this problem is x_0=(0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5). The proposed method solves this problem via the following steps:

1) Starting from x_0, TT-IPM finds a local optimal solution x_s^1=(0.9799, 0.9845, 0.9835, 0.1740, 0.9998, 0.9996, 0.7576, 0.0004, 0.9998, 0.9998, 0.0685) with objective $f(x, y) = -0.9488$. Around x_s^1, there is no higher tier local optimal solution found. The merit values computed at x_s^1 are η_1^+=(4.9548, -4.7×10^{-6}, 0.0011, 1.4430, 3.9928, 3.2×10^{-5}, 0.0005, 5.6004) and η_1^-=(0.0078,0.0313,0.0369,0.0339,1.0×10^{-5},0.5122,1.9253,0.0034). Hence, $y_2 = 1$ and x_0^1=(0.9799, 0.9845, 0.9835, 0.1740, 1, 0.9996, 0.7576, 0.0004, 0.9998, 0.9998, 0.0685).

2) Starting from x_0^1, TT-IPM finds the local optimal solution x_s^2=(0.9799,

0.9845, 0.9835, 0.1748, 1, 0.9997, 0.7568, 0.0004, 0.9998, 0.9999, 0.0685) with objective $f(x,y) = -0.9488$. Around x_s^2, there is no higher tier local optimal solution found. The merit values computed at x_s^2 are η_1^+=(4.9503, 0.0009, 1.4480, 3.9931, 2.7×10^{-5}, 0.0004, 5.6003) and η_1^-=(0.0078, 0.0369, 0.0339, 8.9×10^{-6}, 0.5141, 1.9252, 0.0033). Hence, $y_5 = 0$ and x_0^2=(0.9799, 0.9845, 0.9835, 0.1748, 1, 0.9997, 0.7568, 0, 0.9998, 0.9999, 0.0685).

3) Starting from x_0^2, TT-IPM finds the local optimal solution x_s^3=(0.9799, 0.9845, 0.9835, 0.1739, 1, 0.9997, 0.7579, 0, 0.9998, 0.9999, 0.0685) with objective $f(x,y) = -0.9488$. Around x_s^3, there is no higher tier local optimal solution found. The merit values computed at x_s^3 are η_1^+=(4.9554, 0.0010, 1.4415, 2.8×10^{-5}, 0.0005, 5.6003) and η_1^-=(0.0078, 0.0369, 0.0342, 0.5125, 1.9252, 0.0033). Hence, $y_6 = 1$ and x_0^3=(0.9799, 0.9845, 0.9835, 0.1739, 1, 0.9997, 0.7579, 0, 1, 0.9999, 0.0685).

4) Starting from x_0^3, TT-IPM finds the local optimal solution x_s^4=(0.9799, 0.9845, 0.9835, 0.1739, 1, 0.9997, 0.7578, 0, 1, 0.9999, 0.0685) with objective $f(x,y) = -0.9488$. Around x_s^4, there is no higher tier local optimal solution found. The merit values computed at x_s^4 are η_1^+=(4.9554, 0.0009, 1.4419, 0.0004, 5.6003) and η_1^-=(0.0078, 0.0369, 0.0339, 1.9252, 0.0033). Hence, $y_7 = 1$ and x_0^4=(0.9799, 0.9845, 0.9835, 0.1739, 1, 0.9997, 0.7578, 0, 1, 1, 0.0685).

5) Starting from x_0^4, TT-IPM finds the local optimal solution x_s^5=(0.9799, 0.9845, 0.9835, 0.1746, 1, 0.9997, 0.7570, 0, 1, 1, 0.0686) with objective $f(x,y) = -0.9488$. Around x_s^5, there is no higher tier local optimal solution found. The merit values computed at x_s^5 are η_1^+=(4.9515, 0.0008, 1.4419, 5.6000) and η_1^-=(0.0078, 0.0369, 0.0339, 0.0031). Hence, $y_3 = 1$ and x_0^5=(0.9799, 0.9845, 0.9835, 0.1746, 1, 1, 0.7570, 0, 1, 1, 0.0686).

6) Starting from x_0^5, TT-IPM finds the local optimal solution x_s^6=(0.9799,

0.9845, 0.9835, 0.1739, 1, 1, 0.7576, 0, 1, 1, 0.0684) and $f(x,y) = -0.9488$. Around x_s^6, there is no higher tier local optimal solution found. The merit values computed at x_s^6 are η_1^+=(4.9552, 1.4432, 5.6011) and η_1^-=(0.0078, 0.0339, 0.0031). Hence, $y_8 = 0$ and x_0^6=(0.9799, 0.9845, 0.9835, 0.1739, 1, 1, 0.7576, 0, 1, 1, 0).

7) Starting from x_0^6, TT-IPM finds the local optimal solution x_s^7=(0.9816, 0.9858, 0.9800, 0.2130, 1, 1, 0.7870, 0, 1, 1, 0) and $f(x,y) = -0.9483$. Around x_s^7, there is no higher tier local optimal solution found. The merit values computed at x_s^7 are η_1^+=(4.7194, 1.2692) and η_1^-=(0.0087, 0.0322). Hence, $y_1 = 0$ and x_0^7=(0.9816, 0.9858, 0.9800, 0, 1, 1, 0.7870, 0, 1, 1, 0).

8) Starting from x_0^7, TT-IPM finds the local optimal solution x_s^8=(0.9700, 0.9925, 0.9800, 0, 1, 1, 1.0000, 0, 1, 1, 0) and $f(x,y) = -0.9435$. Around x_s^8, there is no higher tier local optimal solution found. The merit values computed at x_s^8 are $\eta_1^+ = -1.6 \times 10^{-7}$ and η_1^-=0.0213. Hence, $y_4 = 1$ and x_0^8=(0.9700, 0.9925, 0.9800, 0, 1, 1, 1, 0, 1, 1, 0).

9) Starting from x_0^8, TT-IPM finds the local optimal solution x_s^9=(0.9700, 0.9925, 0.9800, 0, 1, 1, 1, 0, 1, 1, 0) and $f(x,y) = -0.9435$.

Hence the final solution to the optimization problem (5.8) is (x^{opt}, y^{opt}) =(0.9700, 0.9925, 0.9800, 0, 1, 1, 1, 0, 1, 1, 0) and $f(x^{opt}, y^{opt}) = -0.9435$.

5.5 Summary

Many practical applications can be formulated as mixed integer nonlinear programs. In this chapter, we have developed a two-stage TRUST-TECH-based

methodology to systematically compute all the local optimal solutions for general MINLP problems. The methodology consists of two distinct stages: Stage I, for a given MINLP problem, constructs a new continuous constrained problem through relaxing the integral variable, and it searches for all the local optimal solutions of the relaxed problem. Stage II defines a reduced constrained problem for each local optimal solution found in Stage I and computes all the local optimal solutions for the reduced problem, from which the optimal solution of the original MINLP problem is determined.

From a computational viewpoint, for general constraint MINLP problems, it may well be due to the lack of computational methods for computing the complete set of decomposition points that the TRUST-TECH-based methodology can only compute multiple local optimal solutions for the associated unconstrained NLP problem and the corresponding reduced constrained NLP problems. In other words, the TRUST-TECH-based methodology can only compute multiple local optimal solutions, instead of all the local optimal solutions for general MINLP problems.

CHAPTER 6

ELITE: ENSEMBLE OF OPTIMAL, INPUT-PRUNED NEURAL NETWORKS USING TRUST-TECH

6.1 Introduction

Two well-known challenging tasks in the area of machine learning using *artificial neural networks (ANNs)* are the task of network architecture selection and the task of optimal weight training. In deciding the architecture for the *multi-layer perceptron (MLP)*, a large network usually provides better approximation accuracy on (training) data at the cost of generalization capability for the unseen (testing) data [77, 81, 86]. Ensemble offers an effective way to alleviate the burden of tuning the parameters of a single ANN and usually results in improved generalization capability [142, 190]. Several factors that have a direct impact on the ensemble quality such as the accuracy and diversity of member networks [14, 131, 173] and the optimal scheme for combination [76, 165].

We propose a systematic methodology for *Ensemble of Optimal, Input-Pruned Neural Networks Using TRUST-TECH*, termed *ELITE*. There are four stages in ELITE designed to achieve high-performance accuracy and diversity of member networks and to achieve optimal combination of selected member networks. In order to construct high-quality neural network ensembles, a diverse population (member neural networks) is produced using different feature subsets for different members, while accurate individual networks are achieved via optimal training. The global optimizer used in ELITE is based on TRUST-TECH and it plays a critical role in achieving both the optimal training and the optimal combination of member neural networks.

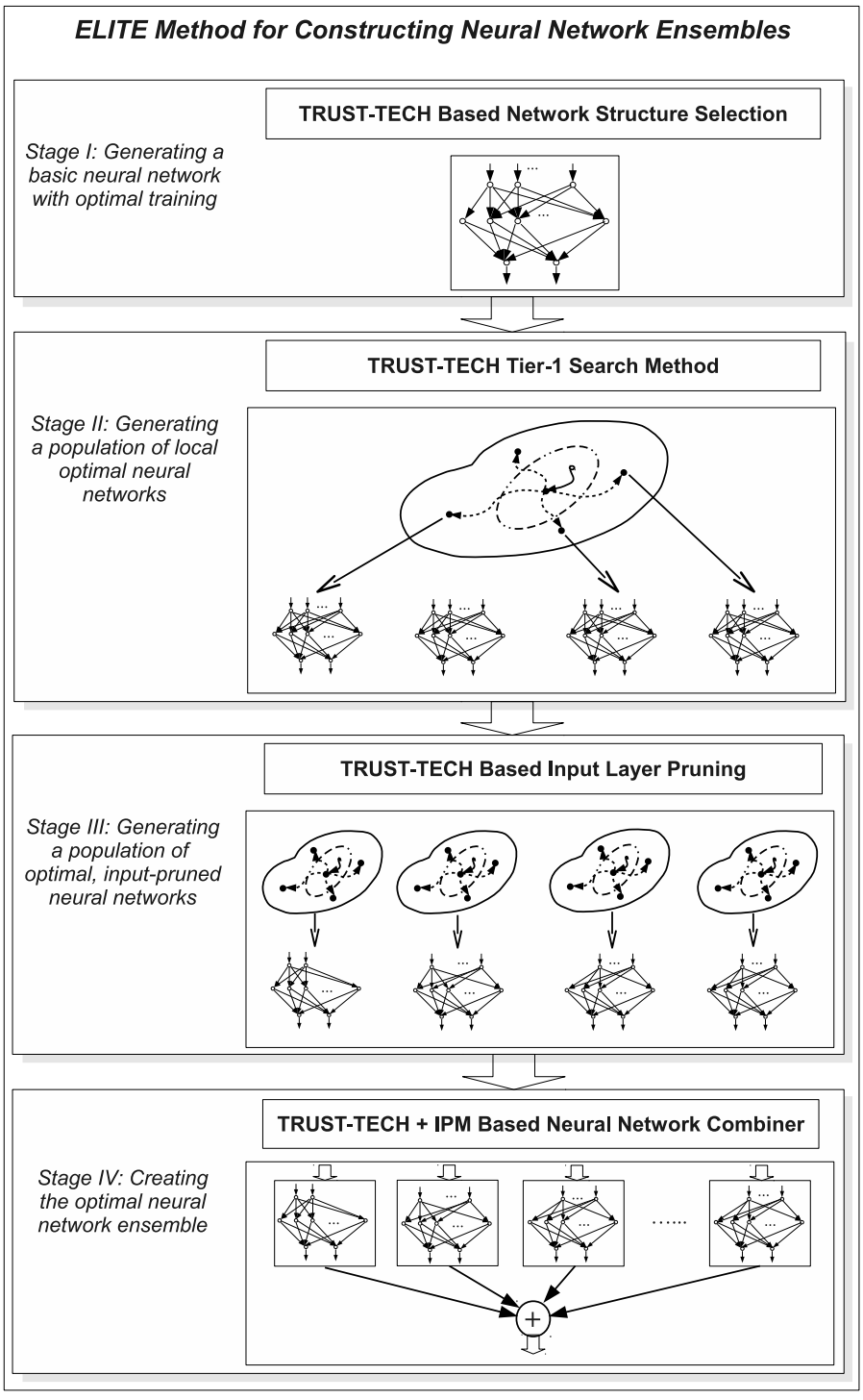


Figure 6.1: Structure of the ELITE method for constructing neural network ensembles.

ELITE provides an effective framework for constructing high-quality neural network ensembles (see Fig. 6.1). Existing training methods for neural networks and methods for composing ensembles can be easily incorporated into ELITE. For instance, in ELITE, optimization problems associated with training and linearly combining neural networks are solved using TRUST-TECH and existing local methods. As illustrated in Fig. 6.1, ELITE creates a population of neural networks through tier-1 TRUST-TECH search. The input layer of each network is pruned and a distinct feature subset is assigned. The accuracy of the input-pruned networks is then achieved by the TRUST-TECH based optimal training. Finally, a combination of TRUST-TECH with interior point method (IPM) is used to compute the optimal (combination) weights and the member neural networks are combined to realize the ensemble.

Several distinguished features of ELITE are described as follows:

1) Diversity and accuracy: Each member neural network in the ensemble constructed by ELITE is associated with a distinct, salient feature subset and is optimally trained with TRUST-TECH. Hence, both accuracy and diversity of the population can be achieved. The deterministic feature selection and TRUST-TECH based optimal training distinguish ELITE from existing methods where feature subsets are randomly generated via bagging methods [157].

2) Optimality: Optimality of the ensemble constructed using ELITE is achieved by optimally combining the member neural networks. The associated quadratic programming problem is effectively solved using TRUST-TECH and IPM. This feature differs from many existing method such as the genetic algorithm based method [190] in that the issue of entrapment in a local optimum is resolved in an efficient and deterministic way.

To illustrate effectiveness of the ELITE method, numerical experiments and studies are carried out for pattern classification using ensemble of feed-forward networks. These numerical experiments and studies are quite extensive and include:

(a) Ensemble performance on a synthetic dataset and several UCI benchmark datasets;

(b) Comparison of ensemble performance by different combination schemes;

(c) Comparison of diversity and accuracy by different schemes with and without using TRUST-TECH;

(d) Ensemble performance with different hidden layer sizes;

(e) Comparison of performance of the proposed ELITE method with existing ensemble methods.

Numerical results show that ELITE consistently outperforms existing methods on the benchmark datasets. The performance of ELITE was compared with that of six existing methods whose performance has been reported in the literature on the same datasets. Of a total of twelve datasets, ELITE achieves the best performance on seven datasets while, on the other five datasets its performance is also comparable with the best performance.

6.2 Preliminaries

6.2.1 Neural network training

The performance of a neural network is usually gauged by measuring *the mean square error (MSE)* of its output. The goal of optimal training is to find a set of parameters that achieves the global minimum MSE [154]. For an n-dimensional dataset, the MSE over Q samples in the training set is given by:

$$E(\mathbf{w}) = \frac{1}{Q}\sum_{i=1}^{Q}\left[t_i - y(\mathbf{x}_i, \mathbf{w})\right]^2, \tag{6.1}$$

where, t_i is the target output for the i-th sample $\mathbf{x}_i$, $\mathbf{w}$ is the weight vector, and $y(\cdot)$ is the network output function. The MSE as a function of the network parameters usually contains many local optimal solutions.

Existing training methods can be categorized into *local* and *global* methods. Several successful training algorithms have been extensively studied in the literature [11, 77, 146]. Local methods, such as the *back-propagation (BP) algorithm*, are usually deterministic and have received significant attention. However, these methods can only attain a local optimal solution close to the initial conditions [69]. On the other hand, global methods, such as *simulated annealing* [4] and *evolutionary algorithms* [90, 109, 182], aim to explore the entire error surface to find solutions approximate to the global optimum. Global methods can explore the entire solution space effectively to identify promising regions [123]. However, these methods may lack the ability to obtain a precise final solution and generally require local methods for fine-tuning.

6.2.2 Neural network based feature selection

The task of feature selection involves selecting a subset of relevant features to build robust learning models. High dimensional features may degrade the efficiency of learning algorithms, especially when irrelevant or redundant features exist [144]. Feature selection has been recognized as a challenging combinatorial optimization problem. It is generally computationally prohibitive to evaluate all possible combinations to find the most compact feature set. As a recent advance, semi-supervised feature selection, where both labelled and unlabelled examples are presented, has attracted special interest. In [176], Xu *et al* solved this problem using convex-concave optimization with encouraging results.

Neural networks can also be used as feature selectors. Neural network based feature selection methods can be categorized as *model independent* or *model dependent* [108]. Model independent methods perform feature selection and model building separately, while model dependent methods attempt to optimize feature selection and model selection simultaneously. Optimal training plays an important role in both model-dependent and model-independent methods when neural networks are used as the feature selector.

6.2.3 Neural network ensemble

An ensemble provides an effective way to alleviate the burden of tuning the parameters of a single learning model. When a number of learning models is available, the best individual is usually chosen. However, the best model with respect to the performance on the training and validation sets does not necessarily have the best performance on the testing set. An ensemble formed by prop-

erly combining the outputs of different models usually results in better generalization performance than any of the involved individuals [64, 142, 183, 190].

Accurate and diverse neural networks are prerequisites to constructing a high-quality ensemble [14, 131, 173]. *Bagging* and *boosting* are two popular methods for creating diverse learning models by altering the training set that each model sees [130]. Bagging is parallel and it re-samples the training set independently for learning each model. In contrast, boosting is sequential and the training set for each model is generated depending on previously learned models. In each case, optimal training is critical to achieve accurate learning models.

The combination scheme has a direct impact on the ensemble quality [155]. Linear combination of neural networks is widely used in constructing an ensemble. The task of getting the optimal combination weights to achieve the minimum error has been formulated as an optimization problem [76, 128, 165]. One difficult issue in the ensemble is the entrapment in local optimal solutions. In the past, different global optimization methods were employed to address this issue with different degree of success. In ELITE, this issue is resolved by a combination of TRUST-TECH and interior point method (IPM).

6.3 Training ANNs Using TRUST-TECH

This section presents an overview of the TRUST-TECH based optimal training method. Without loss of generality, we consider a feed-forward neural network with one input layer, one hidden layer and one output node. Given the input-output pairs $(\mathbf{x}_1, t_1), (\mathbf{x}_2, t_2), \cdots, (\mathbf{x}_Q, t_Q)$, the training task can be formulated as an

s-dimensional optimization problem

$$\min_{\mathbf{w}} E(\mathbf{w}) \tag{6.2}$$

where $s = (n + 2)k + 1$ with n being the number of input nodes and k being the number of hidden nodes, and the weight vector is

$$\mathbf{w} = (w_{01}, \cdots, w_{0k}, \cdots, w_{n1}, \cdots, w_{nk}, b_0, \cdots, b_k)^T,$$

which includes all the network weights (w_{0j} is the j-th output weight, and w_{ij} is the weight connecting i-th input node and j-th hidden node) and biases (b_j is the j-th bias). The MSE to be minimized can be written as

$$E(\mathbf{w}) = \frac{1}{Q} \sum_{i=1}^{Q} [t_i - y(\mathbf{x}_i, \mathbf{w})]^2. \tag{6.3}$$

As we know, TRUST-TECH solves an optimization problem by first defining a dynamical system such that the stable equilibrium points (SEPs) in the dynamical system have one-to-one correspondence with local optimal solutions of the optimization problem (6.2). Because of such correspondence, the problem of computing multiple local optimal solutions of the optimization problem is then transformed to finding multiple stability regions in the defined dynamical system, each of which contains a distinct SEP. A SEP can be computed with the trajectory method or using a local method with a trajectory point in its stability region as the initial point [32, 104].

To solve the optimization problem (6.2), the desired dynamical system can be defined as the generalized gradient system:

$$\frac{d\mathbf{w}}{dt} = -grad_R E(\mathbf{w}) = -R(\mathbf{w})^{-1} \cdot \nabla E(\mathbf{w}), \tag{6.4}$$

where $R(\mathbf{w})$ is a positive definite symmetric matrix (also known as the *Riemannian metric*). This framework is quite general since different existing training

algorithms can be included in this formulation with different definitions of $R(\mathbf{w})$. If $R(\mathbf{w}) = I$, then it is a naive error back-propagation algorithm; if $R(\mathbf{w}) = J(\mathbf{w})^T J(\mathbf{w})$ then it is the Gauss-Newton method; and if $R(\mathbf{w}) = J(\mathbf{w})^T J(\mathbf{w}) + \mu I$ then it will be the Levenberg-Marquardt (LM) method. Hence, TRUST-TECH based methods are dynamical methods for obtaining a set of local optimal solutions of general optimization problems.

A TRUST-TECH based algorithm for training neural network weights is detailed as follows.

The tier-1 TRUST-TECH search based training

Input: a local optimal weight vector $\mathbf{w}_{s0}$.

Output: a set $\mathbf{W}_s$ of next-tier SEPs.

Initialization: $\mathbf{W}_s = \{\mathbf{w}_{s0}\}$

Algorithm:

1) Determine the search directions $\vec{\mathbf{d}}_1, \vec{\mathbf{d}}_2, \cdots, \vec{\mathbf{d}}_k$.
2) $for\ i = 1 : k$
   - Search for an exit point $\mathbf{w}_e$ along $\vec{\mathbf{d}}_i$.
   - if $\mathbf{w}_e$ along the search direction is found, *then*
     - Step forward along $\vec{\mathbf{d}}_i$ to the point $\mathbf{w}' = \mathbf{w}_{s0} + \epsilon(\mathbf{w}_e - \mathbf{w}_{s0})$ with ϵ being a small value. $\mathbf{w}'$ will lie in the stability region of the neighbouring SEP.

- Using $\mathbf{w}'$ as the initial guess, apply the local optimizer to get a tier-1 SEP, denoted as $\mathbf{w}_{si}$, lying in the neighbouring stability region.
- Update $\mathbf{W}_s$ as $\mathbf{W}_s = \mathbf{W}_s \cup \{\mathbf{w}_{si}\}$.

3) Output the set $\mathbf{W}_s$ of SEPs of the generalized gradient system (6.4).

The value of ϵ is empirically chosen to be 0.1 to make $\mathbf{w}'$ in the stability region of the neighbouring stable equilibrium point (or the neighbouring local optimal solution). The task of selecting proper search directions in an efficient way is very challenging. In this algorithm, the search directions can be chosen as a subset of dominant eigenvectors of the objective Hessian at the SEP. Justification of this strategy is that local stable (unstable) manifold of an equilibrium point of (6.4) is tangent to the stable (unstable) eigen-space of the linearized system at this equilibrium point [73]. However, computing Hessian eigenvectors, even dominant ones, is computationally demanding, especially for large-scale problems. Another choice is to use random search directions, but they need be orthogonal to each other in order to span the search space and to maintain a diverse search. It appears that effective directions in general have a close relationship with the structure of the objective function (and the feasible set for constrained problems). Hence, exploitation of the structure of the objective under study will prove fruitful in selecting search directions.

By exploring the TRUST-TECH's capability to escape from local optimal solutions in a systematic and deterministic way, it becomes feasible to locate multiple local optimal solutions in a tier-by-tier manner. Each local optimal solution corresponds to a local optimal neural network. As a result, distinct local opti-

mal neural networks can be obtained (local optimal weights in $\mathbf{W}_s$ of the same network structure) and are denoted as $\mathbf{N} = \{\mathbf{n}_1, \mathbf{n}_2, \cdots, \mathbf{n}_L\}$.

6.4 Optimal Ensemble

This section proposes a method called ELITE for constructing high-quality neural network ensembles, taking advantage of TRUST-TECH's ability to find multiple local optimal solutions. The goals of designing ELITE are two-folds. The first one is to generate a population of accurate and diverse neural networks, and the second one is to optimally combine them to realize an optimal ensemble. ELITE consists of the following four stages (see Fig. 6.2):

- Stage I: Determine an optimal network structure;
- Stage II: Generate member neural networks;
- Stage III: Prune and feature selection for member neural networks; and
- Stage IV: Perform the optimal combination of member neural networks.

6.4.1 Stage I: determining an optimal network structure

Since TRUST-TECH can effectively find multiple local optimal solutions to the training problem., the potential (i.e. the capability) of a neural network with a specific structure can be well explored. Hence, a compact-structure neural network can be obtained. Considering that complexity has a direct impact on the generalization capability of neural networks, a compact-structure neural network is needed. This stage serves to meet this need.

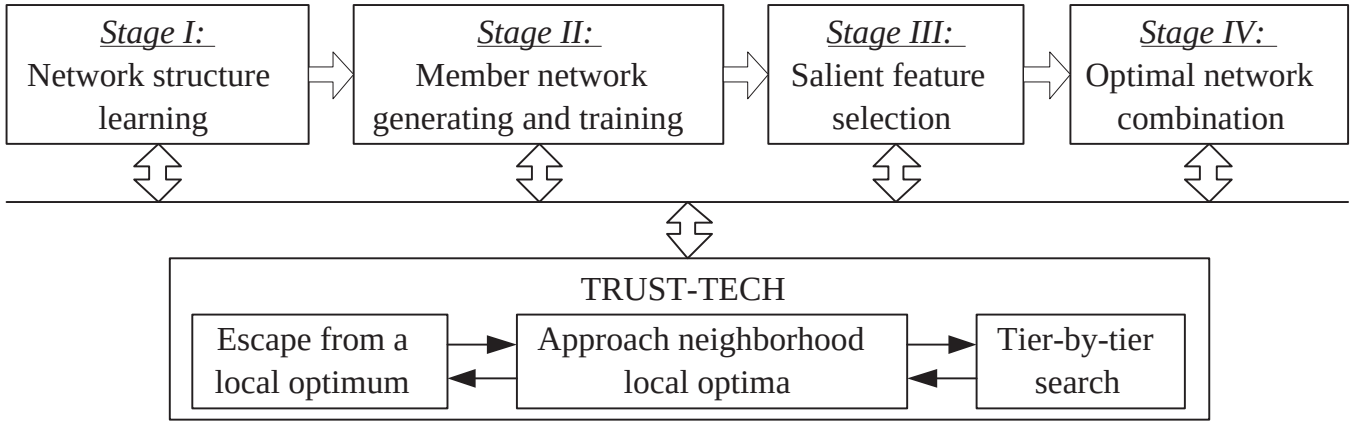


Figure 6.2: ELITE consists of four stages for constructing an ensemble, by generating and optimally combining accurate and diverse neural networks. TRUST-TECH plays a central role in helping the local optimizers avoid entrapment in local optimal solutions to the associated optimization problems.

An incrementally growing method is used to determine a compact-structured neural network. Similar constructive strategy had been used to train feed-forward neural networks with promising results [114]. This method starts from an initial network with a single hidden layer and a small number of hidden nodes. The TRUST-TECH based training method is applied to determine the optimal weights achieving the minimum MSE value. If this value is greater than a target value (0.01 in this chapter), then a new hidden node is added and the network is trained again using TRUST-TECH. This process is repeated until the required MSE value is met or there is no significant improvement can be achieved in reducing the minimum MSE value. The neural network (with high-quality local optimal weights) thus obtained serves as the fundamental neural network for the subsequent stages.

6.4.2 Stage II: generating member neural networks

After the structure of a compact neural network has been determined, the network weights will be re-trained through a tier-1 TRUST-TECH search algorithm. The objective is twofold: First, from a nonlinear system theory viewpoint, this re-training will explore the stability regions of neighbouring SEPs surrounding the SEP corresponding to the basic network. Hence, the possibility of getting better local optimal solutions is increased. Second and more importantly, multiple local optimal solutions will be obtained through the TRUST-TECH search, providing a population of neural networks to form an ensemble.

As a result, a population of neural networks sharing the same structure but with different local optimal weights is obtained. This set of neural networks is

denoted as $\mathbf{N}_0 = \{(\mathbf{n}_0, \mathbf{w}_1^*), (\mathbf{n}_0, \mathbf{w}_2^*), \cdots, (\mathbf{n}_0, \mathbf{w}_L^*)\}$. In our previous work reported in [37], the effort was stopped here and the neural network achieving the best performance on the validation dataset was selected as the final result. However, the single neural network thus selected does not necessarily have the best performance on an unseen testing dataset. In other words, the generalization capability of the selected neural network is not guaranteed. In the proposed ELITE method, all these tier-1 neural networks are preserved. To achieve improved generalization performance, additional processes including feature selection and network ensemble will be carried out, as described in the next two stages.

6.4.3 Stage III: input pruning and population diversity

In this stage, an improved saliency-based method is proposed to achieve optimal input-pruned neural networks using the TRUST-TECH based training method.

In the MLP, each feature is associated with an input node and feature selection can be realized based on evaluating the saliency of the input nodes. The saliency of a weight in the neural network can be approximated by the change on the performance caused by adjusting this weight to 0. Applying the Taylor expansion on the error function $E(\mathbf{w})$ with respect to the i-th weight $w_i \in \mathbf{w} = (w_{01}, \cdots, b_k)^T$, $i = 1, \cdots, s$, we can get

$$\Delta E = \left(\frac{\partial E}{\partial w_i}\right)^T \cdot \Delta w_i + \frac{1}{2}\frac{\partial^2 E}{\partial w_i^2} \cdot \Delta w_i^2 + O(|\Delta w_i|^3). \tag{6.5}$$

In order to adjust w_i to 0, we have $\Delta w_i = -w_i$; meanwhile, the higher order item

in (6.5) can be approximated with

$$O(|\Delta w_i|^3) \approx \frac{1}{2}\frac{\left(\frac{\partial E}{\partial \omega_i}\right)^2}{\frac{\partial^2 E}{\partial \omega_i^2}}. \tag{6.6}$$

Hence, we have the following representation for the saliency of the i-th weight in the neural network:

$$s_i = -\frac{\partial E}{\partial \omega_i}\omega_i + \frac{1}{2}\frac{\partial^2 E}{\partial \omega_i^2}\omega_i^2 + \frac{1}{2}\frac{\left(\frac{\partial E}{\partial \omega_i}\right)^2}{\frac{\partial^2 E}{\partial \omega_i^2}}. \tag{6.7}$$

Consequently, the saliency for an input node (accordingly, the corresponding input feature) can be represented as [108]:

$$S_i = \sum_{j \in fanout\{i\}} \left\{ -\frac{\partial E}{\partial \omega_j}\omega_j + \frac{1}{2}\frac{\partial^2 E}{\partial \omega_j^2}\omega_j^2 + \frac{1}{2}\frac{\left(\frac{\partial E}{\partial \omega_j}\right)^2}{\frac{\partial^2 E}{\partial \omega_j^2}} \right\}. \tag{6.8}$$

In essence, the saliency of an input node is the accumulated saliency of its fan-out weights.

We describe a saliency-based feature selection procedure as follows.

The saliency based method for feature selection

Input: a local optimal network $\mathbf{n}$ and the threshold p.

Output: a salient feature subset $\overline{\mathbf{x}}$, and the input-pruned network $\overline{\mathbf{n}}$.

Algorithm:

1) Calculate the input node saliency according to (6.8).

2) Calculate the normalized saliency

$$\tilde{S}_i = \frac{S_i}{\sum_{i=1}^{n} S_i}, \ \forall i = 1, \cdots, n. \tag{6.9}$$

3) Sort $\{\tilde{S}_i\}$ in the descending order, resulting

$$\{\tilde{S}_{i_j} | \tilde{S}_{i_1} \geq \tilde{S}_{i_2} \geq \cdots \geq \tilde{S}_{i_n}\}. \tag{6.10}$$

4) The selected salient feature subset is:

$$\overline{\mathbf{x}} = \{x_{i_1}, \cdots, x_{i_k}\}, \tag{6.11}$$

where, k is determined via

$$k = \max\{k \mid 1 - \sum_{j=1}^{k} \tilde{S}_{i_j} \geq p\}. \tag{6.12}$$

5) Remove the redundant input nodes and corresponding weights and get the input-pruned network $\overline{\mathbf{n}}$.

The saliency of input nodes is computed. Then the nodal saliency is normalized according to the total saliency. To balance the number of selected features, the saliency threshold p is chosen empirically as 0.15 in this chapter. In other words, the minimum set of input nodes (or features) whose saliency combined accounts for at least 85% of the total saliency is selected.

This feature selection procedure is carried out separately on each tier-1 local optimal neural network in the set $\mathbf{N}_0$. As a result, distinct feature subsets will be assigned to different local optimal neural networks. Since those low-saliency nodes have been removed, the input layer is condensed and the network structure is modified accordingly. These input-pruned neural networks are denoted as $\{\overline{\mathbf{n}}_i : \overline{\mathbf{w}}_i, \overline{\mathbf{x}}_i\}$, $i = 1, \cdots, L$, where $\overline{\mathbf{n}}_i$ stands for the corresponding network structure, $\overline{\mathbf{w}}_i$ corresponds to the modified weights, and $\overline{\mathbf{x}}_i$ is the dataset composed of the selected feature subset.

For each structurally modified neural network $\overline{\mathbf{n}}_i$, it is evident that the remaining weights $\overline{\mathbf{w}}_i$ do not necessarily remain (local) optimal. The TRUST-TECH based training method is carried out to find multiple local optimal weight vectors for each neural network, from which the optimal one $\overline{\mathbf{w}}_i^*$ will be selected. As a result of this stage, a population of optimal networks associated with different feature subsets is obtained and will be denoted as $\{\overline{\mathbf{n}}_i : \overline{\mathbf{w}}_i^*, \overline{\mathbf{x}}_i\}$, $i = 1, \cdots, L$.

6.4.4 Stage IV: optimal combination

The task of finding an optimal ensemble of a family of neural networks is achieved by solving the following optimization problem:

$$\min_{\mathbf{v}} E(\mathbf{v}|\mathbf{N}, \mathbf{x}) = \sum_{i=1}^{Q} \Big(\sum_{j=1}^{L} v_j \circ f_j(\mathbf{x}_i) - t_i \Big)^2, \tag{6.13}$$

where, $E(\mathbf{v}|\mathbf{N}, \mathbf{x})$ is the error function with respect to the combination rule $\mathbf{v}$, given the set of neural networks $\mathbf{N} = \{\mathbf{n}_1, \cdots, \mathbf{n}_L\}$. The dataset $\mathbf{x}$ is usually chosen as the validation dataset. $f_j(\mathbf{x}_i)$ is the output of the j-th neural network when the i-th sample is input, while t_i is the desired output.

Optimal linear combination

ELITE combines the neural networks using optimal linear weights. The optimal weights are calculated by solving the quadratic programming (QP) problem [190]:

$$\begin{aligned} \min_{\mathbf{v}} \quad & E(\mathbf{v}) = \tfrac{1}{2}\mathbf{v}^T\mathbf{C}\mathbf{v} \\ s.t. \quad & \mathbf{v}^T\mathbf{e} = 1 \\ & \mathbf{v} \geq 0 \end{aligned} \quad , \tag{6.14}$$

where, $\mathbf{v}$ stands for the combination weight vector and $\mathbf{e} = (1, \cdots, 1)^T$. $\mathbf{C}$ is the correlation matrix whose elements

$$C_{ij} = \int p(x)(f_i(x) - t(x))(f_j(x) - t(x))dx \tag{6.15}$$

are the correlation between the outputs of the i-th and j-th neural networks, where $t(x)$ is the target output for x. For a practical problem with Q training samples, the matrix $\mathbf{C}$ can be numerically evaluated with elements being

$$C_{ij} = \frac{1}{Q}\sum_{k=1}^{Q}(f_i(x_k) - t_k)(f_j(x_l) - t_l),\ i, j = 1, \cdots, L. \tag{6.16}$$

Since $\mathbf{C}$ may not always have positive eigenvalues, the quadratic optimization problem (6.14) is not necessary convex. Hence, it might have multiple local optimal solutions. Iterative methods, such as the *interior-point method (IPM)* [129], are very effective when solving convex quadratic optimizations problems. However, if there are multiple local optimal solutions, they may get stuck in a local optimal solution. ELITE combines TRUST-TECH and IPM to effectively find multiple local optimal solutions, from which a high-quality solution will be selected for ensemble.

The IPM formulation

Using the logarithmic barrier function, the augmented Lagrange function is:

$$L_\mu(\mathbf{v}, \lambda) = \frac{1}{2}\mathbf{v}^T\mathbf{C}\mathbf{v} + \lambda\left(\mathbf{v}^T\mathbf{e} - 1\right) + \mu\sum_{i=1}^{n}\ln v_i, \tag{6.17}$$

where, μ is the barrier parameter. Hence, the KKT optimality conditions are:

$$\frac{\partial L_\mu}{\partial \mathbf{v}} = \mathbf{C}\mathbf{v} + \lambda\mathbf{e} - \mu\mathbf{V}^{-1}\mathbf{e} = 0 \tag{6.18a}$$

$$\frac{\partial L_\mu}{\partial \lambda} = \mathbf{v}^T\mathbf{e} - 1 = 0 \tag{6.18b}$$

, where, $\mathbf{V} = diag(v_1, v_2, \cdots, v_n)$. Multiplying both sides of (6.18a) with $\mathbf{V}$, we have

$$H_\mu(\mathbf{v}, \lambda) = \begin{pmatrix} \mathbf{VCv} + \lambda\mathbf{Ve} - \mu\mathbf{e} \\ \mathbf{v}^T\mathbf{e} - 1 \end{pmatrix} = \mathbf{0}. \tag{6.19}$$

The essence of IPM is to solve a sequence of problem (6.18) with decreasing $\mu \to 0$. The obtained sequence of solutions will approach a local optimal solution to the original quadratic optimization problem (6.14) where $\mu = 0$.

The TRUST-TECH based method

The IPM can only obtain a local optimal solution providing an initial point. TRUST-TECH is used to compute multiple local optimal solutions with the IPM being the local solver. From these solutions the best one is selected and used for constructing the ensemble. To this end, TRUST-TECH first builds an associated nonlinear dynamical system.

Let $\mathbf{x} = (\mathbf{v}, \lambda)$. The generalized gradient system corresponding to the problem (6.19) is defined as

$$\frac{d\mathbf{x}}{dt} = -\nabla H^T(\mathbf{x}) \cdot H(\mathbf{x}), \tag{6.20}$$

and the associated energy function is

$$E(\mathbf{x}) = \frac{1}{2}\|H(\mathbf{x})\|^2. \tag{6.21}$$

Since the system (6.20) is defined for searching multiple local optimal solutions starting from a local optimal solution of problem (6.14), $\mu = 0$ and thus has no presence in (6.20).

The TRUST-TECH based method for calculating the optimal combination weights is detailed as follows.

The TRUST-TECH+IPM algorithm for optimal linear combination

Input: The set of neural networks $\mathbf{N} = \{\mathbf{n}_1, \cdots, \mathbf{n}_L\}$.

Output: The optimal combination weights $\mathbf{v}*$.

Initialization: The initial point $\mathbf{v}_0 = \{1/L, \cdots, 1/L\}$, and the set of SEPs $\mathbf{V}_s = \emptyset$.

Algorithm:

1) Calculate the correlation matrix $\mathbf{C}$ using (6.16) and compute its eigenvalues $\lambda_1, \cdots, \lambda_k$.
2) Using $\mathbf{v}_0$ as the initial point, apply the IPM to solve (6.14) and get an SEP $\mathbf{v}_{s0}$. Update $\mathbf{V}_s$ as $\mathbf{V}_s = \{\mathbf{v}_{s0}\}$.
3) If $\min_{i=1}^{k} \lambda_i < \sigma$ (σ is a small positive value)
   - Calculate the search directions $\{\vec{\mathbf{d}}_1, \cdots, \vec{\mathbf{d}}_m\}$.
   - *for* $i = 1 : m$
     - ⋆ Search for an exit point $\mathbf{v}_e$ along $\vec{\mathbf{d}}_i$ in the generalized gradient system (6.20).
     - ⋆ If $\mathbf{v}_e$ along $\vec{\mathbf{d}}_i$ is found, then
       - · Step forward along the search direction to the point $\mathbf{v}' = \mathbf{v}_{s0} + \epsilon(\mathbf{v}_e - \mathbf{v}_{s0})$ with ϵ being a small positive number. $\mathbf{v}'$ will lie in the stability region of the neighbouring SEP.
       - · Using $\mathbf{v}'$ as the initial point, apply the IPM to get the tier-1 SEP, denoted as $\mathbf{v}_{si}$, lying in the neighbouring stability region.
       - · Update $\mathbf{V}_s$ as $\mathbf{V}_s = \mathbf{V}_s \cup \{\mathbf{v}_{si}\}$.

4) The optimal combination weight vector is: $\mathbf{v}^* = \arg\min_{\mathbf{v}}\{E(\mathbf{v}) \mid \mathbf{v} \in \mathbf{V}_s\}$.

Being the same as in the previous tier-1 TRUST-TECH search for training, the value of ϵ is chosen to be 0.1 in this chapter, to make $\mathbf{v}'$ closer to the neighbouring stable equilibrium point (or the neighbouring local optimal solution). In this method for computing combination weights, σ is used to detect non-convex situations, which is set to $1e-6$ in this chapter.

6.5 Numerical Results

To evaluate the performance of the neural network ensembles constructed using ELITE, numerical experiments have been carried out to solve a variety of pattern classification problems on the synthetic dataset and the UCI benchmark datasets [6]. In this section, these experiments are described in detail and the results are presented and discussed.

6.5.1 Experimental set-up

ELITE has been implemented in MATLAB, and the ensemble is constructed by optimally combining feed-forward networks with one hidden layer. The *hyperbolic tangent function* is used as the transfer function in both the hidden and output layers, and the *Levenberg-Marquardt method* is used as the local optimizer for training.

Starting from the initial local optimal solution obtained by the local optimizer, we apply the TRUST-TECH tier-1 search algorithm to locate the tier-1 local optimal solutions. The search directions from the initial local optimal solution are 20 random but orthogonal directions. Hence, the largest ensemble size will be 21 (the base neural network and 20 tier-1 neural networks). The number of directions is determined based on the result reported in [61], where both the theoretical and experimental results showed that choosing the ensemble size to be between 10 and 20 is sufficient to approach the asymptotic bagging error. In addition, the experimental results in [61] showed that the performance improvement becomes flat when the ensemble size is larger than 20. In ELITE, if multiple networks have the same set of inputs, only the best one will be involved in ensemble. Furthermore, networks with accuracy lower than 50% are removed from combination. As a result, the final ensemble size can be smaller than 21.

After having performed the proposed stages of feature selection and TRUST-TECH based optimal training, a family of optimal, yet diverse neural networks are generated. The ensemble is constructed using a set of optimal linear combination weights computed by solving the quadratic program, where the IPM solver in the IPOPT package [170] is used as the local optimizer.

6.5.2 Experiments on the synthetic dataset

Experiments are first carried out on a 2-dimensional synthetic dataset. The main purpose is to visually demonstrate the diversity between different local optimal neural networks that are computed in ELITE. This dataset is composed of

200 points in the xy-plane that have been picked up randomly within the area $\{(x,y)| -1 < x < 1, -1 < y < 1\}$ following the uniform distribution. Two class labels are assigned to the samples following

$$c = \begin{cases} -1, & x \geq -y \\ +1, & x < -y \end{cases}, \tag{6.22}$$

then a zero-mean Gaussian noise with a standard deviation of 0.2 is added to corrupt the data. A half portion from each class is used for training, and the remaining half is used for testing. The distribution of the synthetic dataset is shown in Fig. 6.3(a). The neural network is composed of 4 nodes in the hidden layer. Since there are only 2 features in the dataset, feature selection is not conducted.

Table 6.1 summarizes the performance of the local optimal neural networks and their ensemble. We have the following observations:

(a) Local optimality and diversity: The initial local optimal network has training error of 4% and testing error 11%. Its classification surface shown in Fig. 6.3(b) indicates that it is slightly over-trained. The performance of different tier-1 local optimal neural networks varies significantly, where the training error varies from 2% to 41% and the testing error varies from 9% to 32%.

(b) Minimum MSE neural network: In terms of the objective function value for the optimization problem (6.2), the best MSE is obtained by the 19-th network, which is 0.08. However, although it achieves the best performance on the training data with the classification error being 2%, this network is not the best in terms of the testing performance, which is 15%. Its classification surface is presented in Fig. 6.3(c), showing that this network is severely over-trained.

Table 6.1: The results on the synthetic data

Network #	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
Training MSE	0.14	0.15	1.64	0.56	0.48	0.36	0.24	0.15	0.32	0.32	0.16
MSE Improvement	-	-7.1%	-1071.43%	-300%	-242.86%	-157.14%	-71.43%	-7.1%	-128.57	-128.57%	-14.28%
Training error	4%	4%	41%	14%	12%	9%	6%	4%	8%	8%	4%
Testing error	11%	15%	32%	15%	20%	11%	14%	10%	10%	11%	12%
Network #	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19***	**20**	**21**	**Ensemble**
Training MSE	0.44	0.32	0.18	0.40	0.64	0.27	0.23	0.08	0.14	0.26	-
MSE Improvement	-214.28%	-128.57%	-28.57%	-185.71%	-357.14%	-92.86%	-64.28%	75%	0%	-85.71%	-
Training error	11%	8%	5%	10%	16%	7%	7%	2%	5%	7%	**7%**
Testing error	9%	12%	22%	13%	18%	11%	16%	15%	11%	10%	**10%**

(c) Improvements via ensemble: The ensemble enhances the generalization performance of the learning model. It results in a good balance between the classification errors on the training and testing datasets, which are 7% and 10%, respectively.

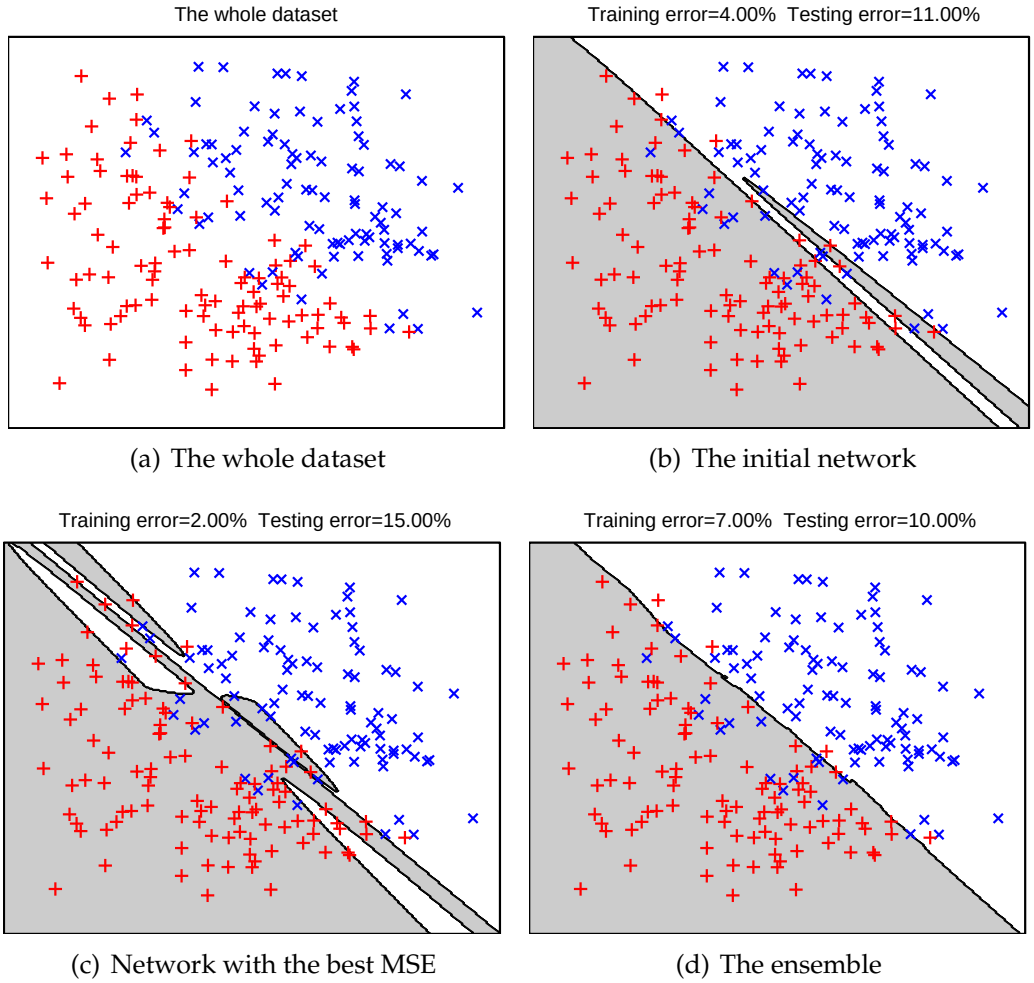
Observation (c) can be further verified by observing its classification surface as shown in Fig. 6.3(d). The classification surface is well structured and very close to the ground-truth classification surface $y = -x$ for the dataset before being contaminated by the noise.

The capability of ELITE to generate a diverse population of neural networks can also be verified by observing the classification surfaces for all the 20 tier-1 local optimal neural networks, as shown in Fig. 6.4. This figure reveals excellent diversity among the local optimal neural networks found by ELITE, even without feature selection. This experiment supports the strategy of ELITE to use different local optimal neural networks as members for ensemble.

6.5.3 Experiments on the UCI benchmark datasets

To evaluate the performance on real data and to facilitate comparison with other existing methods, ELITE has also been tested on the UCI benchmark datasets [6]. 12 datasets for pattern classification have been used in the experiment, which are summarized in Table 6.2. To evaluate the generalization performance, 3-fold cross-validation (CV) is carried out on each dataset.

The number of hidden nodes for each dataset is shown in Table 6.3. This table also summarizes the performance of the tier-1 local optimal neural networks



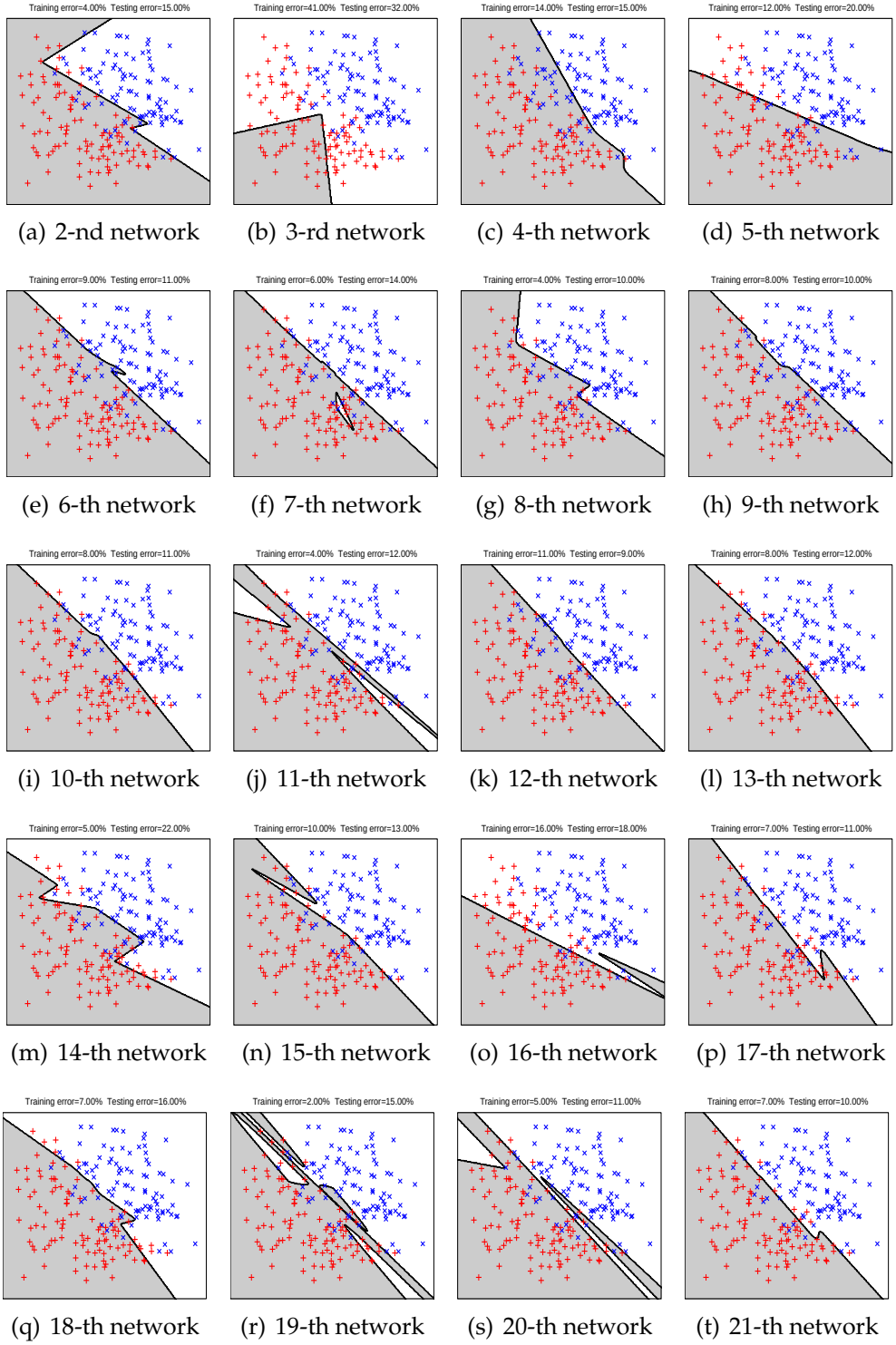
(a) The whole dataset (b) The initial network

(c) Network with the best MSE (d) The ensemble

Figure 6.3: This figure presents the classification surfaces for the initial network, the tier-1 minimum-MSE network and the final ensemble.

obtained by the tier-1 TRUST-TECH search.

Feature selection is carried out on each tier-1 neural network, where the threshold for determining the redundant features is set as $p = 0.15$ in (6.12). The performance of neural networks without feature selections and with feature selection is summarized in Table 6.3 and Table 6.4 respectively. It can be observed that the generalization performance has been greatly improved through feature selection due to a more compact network structure. For example, on average



(a) 2-nd network (b) 3-rd network (c) 4-th network (d) 5-th network

(e) 6-th network (f) 7-th network (g) 8-th network (h) 9-th network

(i) 10-th network (j) 11-th network (k) 12-th network (l) 13-th network

(m) 14-th network (n) 15-th network (o) 16-th network (p) 17-th network

(q) 18-th network (r) 19-th network (s) 20-th network (t) 21-th network

Figure 6.4: This figure depict the classification surfaces for the 20 local optimal networks obtained by tier-1 TRUST-TECH search. Diversity of the classification surfaces is readily observed.

Table 6.2: The datasets used in the experiments

Dataset	Dataset Name	Samples	Features	Classes
1	Breast Cancer	683	9	2
2	Clean	467	168	2
3	Diabetes	768	8	2
4	Glass	214	9	6
5	Ionosphere	351	33	2
6	Iris	150	4	3
7	MAGIC	19020	11	2
8	Segmentation	210	19	7
9	Sonar	208	60	2
10	SPECTF	80	44	2
11	Statlog (Heart)	270	13	2
12	Wine	178	13	3

37.00% features have been eliminated from the *Breast cancer* dataset and the classification error is increased from 0.08% to 0.76% on the training set, while the testing error is decreased from 4.96% to 3.24%. For the *SPECTF* dataset, the procedure of feature selection have removed 35.45% features, and yet the training error is maintained at 0.0% while the testing error is reduced from 28.94% to 19.99%.

Table 6.5 summarizes the performance of the resultant ensemble. By the performance summarized in Table 6.4 and 6.5, it follows that the improvement on performance made by Stage IV of ELITE is quite significant. For example, for the *Breast cancer* dataset, the classification error on the testing set is reduced

Table 6.3: Performance of the tier-1 min MSE neural networks

Dataset	Hidden nodes	Training error		Testing error	
		μ_{Tr}	σ_{Tr}	μ_{Ts}	σ_{Ts}
1	6	0.08%	0.08%	4.96%	0.12%
2	10	0.13%	0.02%	9.67%	0.73%
3	6	13.13%	0.66%	27.04%	0.27%
4	6	18.17%	2.73%	44.14%	1.29%
5	10	0.0%	0.0%	10.13%	0.96%
6	4	0.20%	0.45%	5.39%	1.14%
7	9	12.21%	0.07%	13.23%	0.09%
8	8	1.47%	0.62%	14.57%	0.29%
9	10	0.0%	0.0%	23.87%	0.29%
10	6	0.0%	0.0%	28.94%	1.30%
11	8	2.31%	1.24%	23.20%	0.63%
12	4	0.0%	0.0%	5.46%	0.77%

μ, σ: the mean and standard deviation, respectively

from 3.24% to 1.44%. For the *Wine* dataset, the improvement is more significant, with the testing error being decreased from 4.84% to 0.19%.

Generally speaking, Stage IV of ELITE constructs a high-performance ensemble by optimally ensembles a family of diverse and yet locally optimal neural networks built in Stage I through III of ELITE. For instance, it is observed that, comparing the performance of the solo neural network obtained in the stage I, the ensemble created using ELITE also reduces the standard deviation σ of the classification error. For example, for the *Iris* dataset, σ of the testing

Table 6.4: Performance of the input-pruned neural networks

Dataset	Feature set		Training error		Testing error	
	Size	Reduction	μ_{Tr}	σ_{Tr}	μ_{Ts}	σ_{Ts}
1	5.67	37.00%	0.76%	0.11%	3.24%	0.09%
2	117.2	29.40%	0.10%	0.01%	9.43%	0.94%
3	5.87	26.62%	17.70%	0.57%	24.40%	0.23%
4	6.07	32.56%	30.28%	2.54%	42.85%	1.97%
5	19.07	42.21%	0.09%	0.05%	9.52%	0.68%
6	3.07	23.25%	0.45%	0.16%	2.78%	0.67%
7	6.13	44.27%	13.20%	0.07%	13.53%	0.08%
8	9.47	50.16%	2.18%	0.68%	11.11%	1.78%
9	36.87	38.55%	0.37%	0.11%	17.08%	0.74%
10	28.40	35.45%	0.0%	0.0%	19.99%	1.40%
11	8.33	35.92%	4.08%	1.15%	19.92%	0.59%
12	8.0	38.46%	0.0%	0.0%	4.84%	0.90%

error is 0.59% before using TRUST-TECH, which is increased to 1.14% for the tier-1 best neural network and 0.67% after using the proposed feature selection method. ELITE achieves a significant decrease on σ to 0.13%. The direct benefit introduced by the reduced deviation is that sensitivity of the classifier's performance in initialization can be effectively suppressed. Hence, improvement on both performance and robustness achieved by ELITE has been well demonstrated.

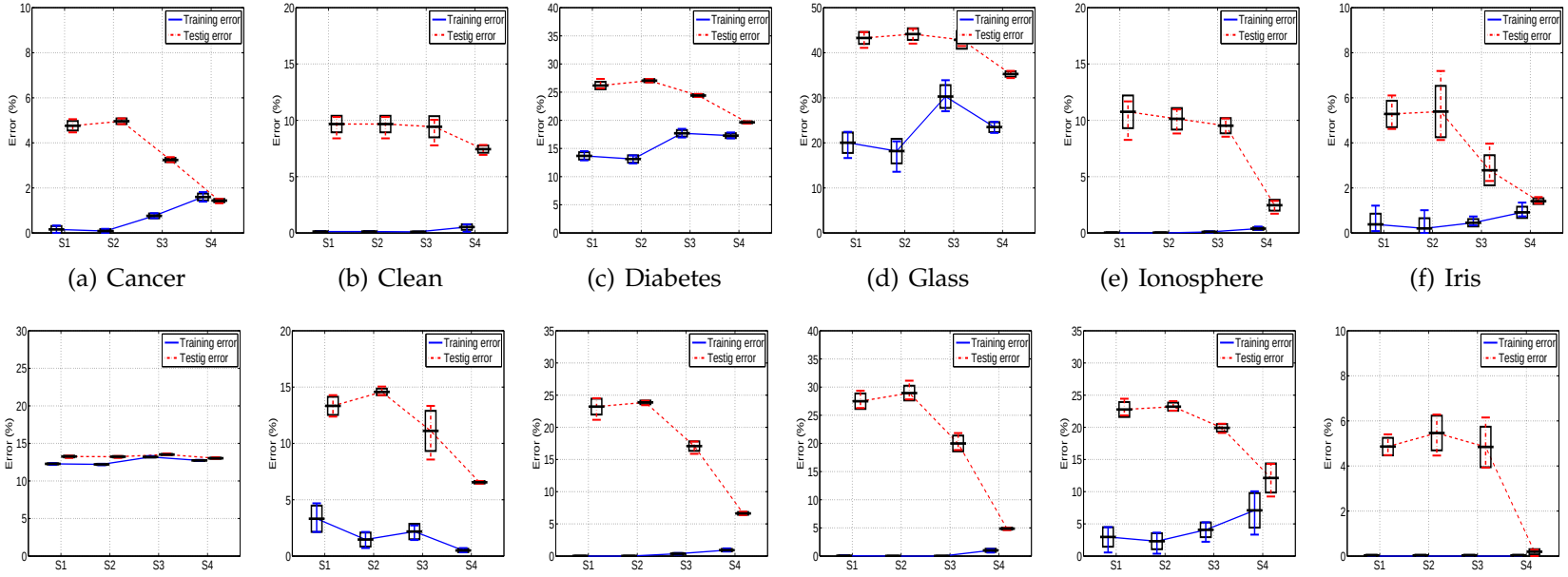
Among all learning iterations involved in the numerical experiments on the UCI datasets, it is found that the condition in the algorithm for computing en-

Table 6.5: Performance of the ensemble

Dataset	Ensemble size		Training error		Testing error	
	μ_{Sz}	σ_{Sz}	μ_{Tr}	σ_{Tr}	μ_{Ts}	σ_{Ts}
1	19.12	0.47	1.60%	0.15%	1.44%	0.07%
2	20.02	0.11	0.53%	0.25%	7.44%	0.32%
3	15.66	0.38	17.28%	0.43%	19.64%	0.16%
4	12.91	0.70	23.50%	1.10%	35.27%	0.60%
5	20.26	0.34	0.39%	0.11%	2.46%	0.48%
6	6.47	0.22	0.92%	0.25%	1.41%	0.13%
7	11.97	0.45	12.73%	0.05%	13.03%	0.08%
8	18.06	0.51	0.50%	0.14%	6.55%	0.11%
9	20.81	0.10	0.92%	0.19%	6.63%	0.22%
10	19.85	0.47	1.02%	0.28%	4.87%	0.17%
11	19.87	1.01	7.10%	2.69%	12.13%	2.26%
12	20.43	0.06	0.0%	0.0%	0.19%	0.11%

semble weights (i.e. $min\{\lambda_1, \cdots, \lambda_k\} < \sigma$) is satisfied for 57.3% of the cases. These cases benefit from TRUST-TECH to acquire the optimal combination weights.

Finally, statistics regarding the ensemble size, that is, the number of neural networks involved in the ensemble for each dataset is also presented in Table 6.5 (columns 2 and 3). For the convenience of comparison, Fig. 6.5 summarizes the performance in different stages on each dataset. A consistent improvement of the performance from each stage of ELITE can be readily observed.



(a) Cancer (b) Clean (c) Diabetes (d) Glass (e) Ionosphere (f) Iris

(g) MAGIC (h) Segmentation (i) Sonar (j) SPECTF (k) Statlog (heart) (l) Wine

Figure 6.5: The performance in stages of ELITE. The mean, standard deviation, range of the error are presented. The labels on x-axis stand for the four stages: *S1* for the base local optimal network obtained in stage 1; *S2* for tier-1 minimum-MSE neural networks in stage 2; *S3* for the optimal input-pruned neural networks in stage 3; and *S4* for the final ensemble in stage 4. Consistent improvement of the performance from stage to stage can be readily observed.

6.5.4 Comparison with other ensemble schemes

We next show the effectiveness of Stage IV of ELITE. To this end, the performance of different ensemble schemes is compared. In the past, ensembles have also been constructed using other two widely adopted schemes: the uniform linear combination and majority voting. On the other hand, the optimal linear combination scheme is employed in ELITE.

A comparison between the performance by these three ensemble schemes is presented in Table 6.6. Using the performance by voting as the baseline, a comparison of the testing error by the three schemes is better visualized in Fig. 6.6. We have the following observations based on the comparison:

Firstly, ELITE outperforms the other two schemes by using the TRUST-TECH based optimal linear combination scheme. On all the datasets, the scheme achieves testing performance better than both the uniform linear combination and majority voting. The benefit of being computationally efficient in finding optimal combination weights to construct high-quality ensembles makes the proposed ELITE method a very favourable choice in practical applications.

Secondly, the uniform linear combination scheme is comparable with the majority voting scheme. In fact, these two ensemble schemes share almost indistinguishable performance when they are used to combine the local optimal neural networks learned in stages I through III of ELITE.

Table 6.6: Comparison between different ensemble schemes

Dataset	ELITE		Uniform weight		Voting	
	E_{Tr}	E_{Ts}	E_{Tr}	E_{Ts}	E_{Tr}	E_{Ts}
1	1.60%	1.44%	1.77%	1.99%	1.70%	1.84%
2	0.53%	7.44%	2.66%	11.19%	2.43%	10.82%
3	17.28%	19.64%	17.69%	21.28%	17.18%	21.46%
4	23.50%	35.27%	21.66%	35.94%	28.53%	37.59%
5	0.39%	2.46%	0.25%	5.09%	0.25%	5.10%
6	0.92%	1.41%	0.93%	2.11%	1.13%	1.63%
7	12.73%	13.03%	13.17%	13.52%	13.57%	13.77%
8	0.50%	6.55%	0.04%	7.45%	1.10%	7.41%
9	0.92%	6.63%	0.14%	10.84%	0.13%	10.84%
10	1.02%	4.87%	0.0%	11.00%	0.0%	10.31%
11	7.10%	12.13%	6.41%	15.01%	6.23%	14.79%
12	0.0%	0.19%	0.0%	0.89%	0.0%	0.59%

6.5.5 Diversity and accuracy

Relationship between the diversity and accuracy of the member networks involved in ensemble is examined. In particular, the relationship exhibited in ELITE is numerically studied. The diversity of a family of neural networks is evaluated as the *averaged double fault measure* [99]:

$$div = 1 - \frac{2}{L(L-1)} \sum_{i=1}^{L-1} \sum_{j=i+1}^{L} prob(\mathbf{n}_i \text{ fails}, \ \mathbf{n}_j \text{ fails}), \tag{6.23}$$

where, $\mathbf{n}_i$ and $\mathbf{n}_j$ are the i-th and j-th member networks, respectively, and L is the total number of member networks.

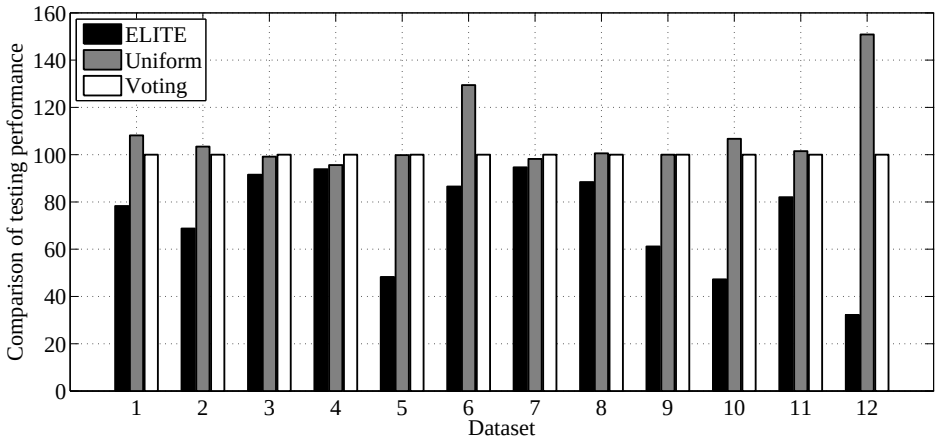


Figure 6.6: Comparison of the testing error of three ensemble schemes. The voting performance is used as the baseline for comparison.

A numerical test is carried out on the *Sonar* dataset. The following six families of neural networks are produced:

(a) fully connected networks training by TRUST-TECH;

(b) fully connected networks with random initialization and trained by a local optimizer without using TRUST-TECH;

(c) input-pruned and retrained networks generated from (a);

(d) input-pruned and retrained networks generated from (b);

(e) input-pruned networks (no retraining) generated from (a);

(f) input-pruned networks (no retraining) generated from (b).

Ensembles are constructed using the proposed optimal combination method and their accuracy is evaluated.

The diversity and accuracy of these ensembles is shown in Table 6.7. We have the following observations based on our numerical results:

Table 6.7: Diversity and accuracy

Scheme	Diversity		Training error		Testing error	
	μ_{div}	σ_{div}	μ_{Tr}	σ_{Tr}	μ_{Ts}	σ_{Ts}
a	0.902	0.007	0.48%	0.18%	10.26%	0.74%
b	0.886	0.008	0.13%	0.09%	12.60%	0.56%
c	0.921	0.007	0.92%	0.19%	6.63%	0.22%
d	0.884	0.004	0.93%	0.21%	10.95%	0.90%
e	0.869	0.009	9.85%	0.39%	13.32%	0.68%
f	0.867	0.003	9.04%	0.87%	14.35%	0.58%

(i) The diversity measure is highly correlated with the ensemble accuracy. In other words, a higher diversity is associated with a better ensemble performance.

(ii) Input-pruned neural networks without retraining lead to in the worst-performed ensemble for both scenarios with and without using TRUST-TECH.

(iii) Retraining the input-pruned networks can improve the ensemble performance, which also outperforms the ensemble using networks without input-pruning.

(iv) Using TRUST-TECH in schemes (a), (c) and (e) results better performance than schemes (b), (d) and (f), respectively.

(v) The proposed ELITE method, corresponding to the scheme (c), achieves the best ensemble performance among the six situations. This validates effectiveness of the four-stage TRUST-TECH based procedures in ELITE.

6.5.6 Performance with hidden layer size

In this experiment, the influence of the hidden layer size of member networks over the ensemble performance is studied. This study is carried out on the *Sonar* data, and the hidden layer size is varied between 2 to 11. Since the hidden layer size is fixed, only stage II through stage IV of the proposed method is involved in constructing the ensembles.

Both the training error and testing error with different number of hidden nodes are shown in Fig. 6.7. It can be observed that as the number of hidden nodes increases, the ensemble performance improves accordingly. In addition, when the hidden layer size is larger than 5, the improvement of the ensemble performance becomes saturated, although slightly improved. Finally, it can be observed that the performance reaches an optimal condition when the hidden nodes is 10. It should be noted that this optimal condition comes with a cost, that is, more computational efforts in the tier-1 searches of Stage II and III.

6.5.7 Comparison with existing methods

Finally, the performance of ELITE is compared with six existing methods whose performance were also reported on (part of) the same datasets as those used in this chapter:

- *SVM*: The support vector machine (SVM) using linear kernel that is implemented in LibSVM [54].
- *TRUST-TECH in [37]:* Our previous work reported in [37], where a multi-tier TRUST-TECH search was used for optimally training neural networks.

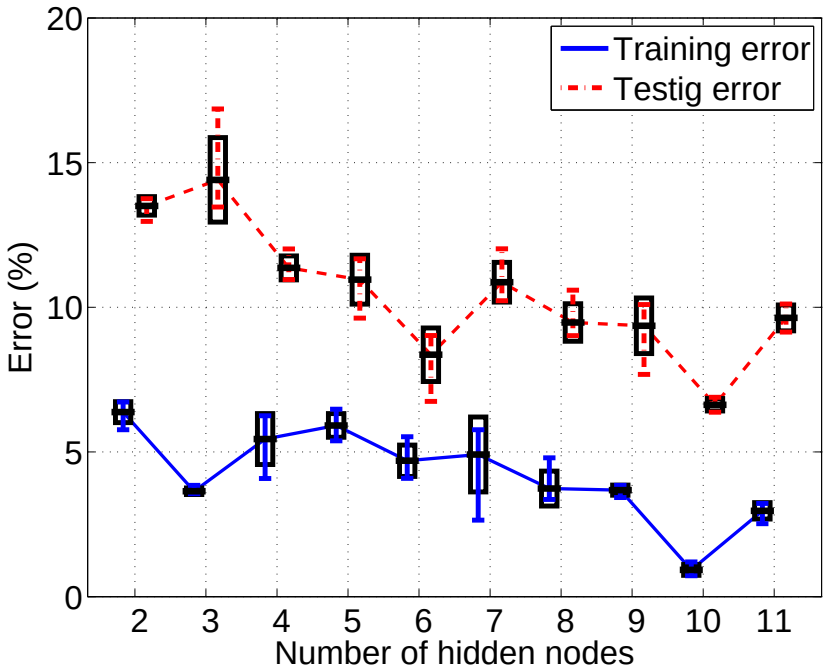


Figure 6.7: Training error and testing error with different number of nodes in the hidden layer. Both the training error and testing error reach their local optimal solutions when the number of hidden nodes is 10.

- *CNNE in [85]:* A constructive algorithm for training cooperative neural network ensembles.
- *COOP in [64]:* The cooperative co-evolutive approach for designing neural network ensembles.
- *Infinite ensemble in [113]:* Infinite ensemble learning using SVMs. The best performance of the ensembles constructed with different kernels that were reported in [113] is considered for comparison.
- *NLBP in [62]:* Ensemble construction based on the use of nonlinear boosting projection. The best performance ensemble of the neural network, the C4.5 tree and SVM that had been reported is considered for comparison.

Table 6.8: Comparison between ELITE and existing methods

Dataset	ELITE	SVM	In [37]	CNNE	COOP	In [113]	NLBP
1	1.44%	3.89%	2.63%	**1.10%**	1.23%	3.11%	28.11%
2	**7.44%**	43.07%	×	×	×	×	×
3	19.64%	24.72%	20.58%	**17.80%**	19.69%	23.50%	23.72%
4	35.27%	31.38%	×	24.60%	**22.89%**	×	31.50%
5	**2.46%**	10.79%	6.54%	×	×	6.40%	4.96%
6	**1.41%**	4.18%	2.67%	×	×	×	4.40%
7	**13.03%**	14.67%	×	×	×	×	×
8	6.55%	17.07%	7.40%	×	×	×	**3.04%**
9	**6.63%**	40.38%	12.98%	14.36%	14.36%	14.70%	15.60%
10	**4.87%**	34.17%	×	×	×	×	×
11	12.13%	29.95%	×	**11.20%**	11.96%	16.40%	18.44%
12	**0.19%**	5.69%	4.48%	×	×	×	×

×: data not available

The comparison results are summarized in Table 6.8. For the convenience of comparison, the best performance on each dataset is highlighted. It can be seen that ELITE achieves the best performance on 7 datasets. On the other 5 datasets, the performance of the ensemble constructed by ELITE is also comparable with the best performance.

Numerical results in this section have shown the effectiveness of ELITE in constructing high-quality neural network ensembles. Specifically, the diversity of the member networks and accuracy of the constructed ensemble have been achieved with the proposed procedures. Hence, ELITE is promising and can be

a competing choice in practical applications.

6.6 Summary

In this work, we have developed a methodology, termed ELITE for constructing high-quality neural network ensembles. ELITE is designed to address the two challenging issues in the area of machine learning using ANNs: the issue of network architecture selection and the issue of optimal weight training. There are four stages in ELITE in which a seed neural network is first constructed (Stage I), followed by a family of member neural networks (Stage II), each of which is optimally pruned (Stage III), and the optimal ensemble is achieved at Stage IV. Several design tasks in ELITE is formulated as optimization problems and solved by the TRUST-TECH method, which provides a systematic and deterministic way to escape from a local optimal solution and to approach multiple local optimal solutions

Distinguished features of ELITE include the following:

1) Diversity: Ensemble members of ELITE are distinct optimal neural networks with different optimally-pruned inputs. These members are generated using the proposed saliency based feature selection method. In this manner, diversity of the ensemble members can be achieved.

2) Accuracy: In ELITE, accuracy of the ensemble members is achieved via selecting high-quality neural networks from multiple local optimal solutions obtained by the TRUST-TECH based training method.

3) Optimality: Optimality of the ensemble in ELITE is achieved by optimally

combining a set of optimal, input-pruned member networks. Specifically, optimality of the member neural networks is attained by using a TRUST-TECH based training method, and by achieving optimality of the combination weights is achieved via solving the associated non-convex quadratic optimization problem using TRUST-TECH and a local optimizer, IPM.

The performance of ELITE was compared with other six methods whose performance has been reported in the literature on the same datasets. Of a total of 12 datasets, ELITE achieves the best performance on 7 datasets, while, on the other 5 datasets its performance is also comparable with the best one. Extensive numerical results conducted thus far have shown the effectiveness of the ELITE method in constructing high-quality neural network ensembles. Hence, ELITE is promising and can be a favourable choice in practical applications.

Further work on improving the generalization performance of the ensemble constructed by ELITE is desirable. Efforts on the following two aspects may prove fruitful:

1) Network weight pruning: In this chapter, feature selection has been implemented as input node pruning. However, the obtained input-pruned neural network is still fully connected between the layers. Hence, there might still be considerable redundancy in network connection. Additional weight pruning to further reduce the network complexity so as to improve diversity of the member networks is desirable.

2) Multi-objective neural network design: The task of network weight training and feature selection have been solved separately. In fact, objectives of these two tasks are generally competing, and it is more suitable to solve them simul-

taneously. To this end, the neural network design method can be formulated as a multi-objective optimization problem and solved by an extended TRUST-TECH method. The development of TRUST-TECH based multi-objective optimization methodology is our another future work.

CHAPTER 7

ELITE-STLF: SHORT-TERM LOAD FORECASTING USING ELITE

7.1 Introduction

Load forecasting is a key component of the daily operation and planning of an electric utility. Forecasts are needed for a variety of activities such as generation scheduling, scheduling of fuel purchase, maintenance scheduling and security analysis [141]. In terms of time stamping, load forecasts required by the power industry can range from short term (a few minutes [27], hours [153], or days ahead [7, 138]) to long term (up to 20 years ahead) [68]. *Short-term load forecasting (STLF)*, in particular, has become increasingly important since the rise of the competitive energy market [78].

Economic and efficient operation of power systems requires accurate load forecasting in order to close track the system load by the system generation at all times. The forecasting error can result in significantly increased operating costs. Under-prediction of load will result in a failure to provide the necessary reserves which will translate to higher costs due to the use of expensive peaking units. Over-prediction of load, on the other hand, will involve the start-up of too many generation units and result in an unnecessary increase in reserves and hence operating costs. In the year 1985, it was estimated that a one-percent increase in the forecasting error was associated with an increase in the operating cost of ten million pounds per year for the British power system [71]. Because of its economic impact, short term load forecasting has become one of the major research areas in electric power engineering.

Accurate load forecasting, however, is a difficult task. First, it is because the load series is complex and exhibits several levels of seasonality. Secondly, it is because there are many important factors, specially weather-related ones, that must be considered in the forecasts [78]. Relationship between these factors and the load forecast has been found to be highly nonlinear. Researches showed that it is relatively easy to construct a forecaster with performance being about 10% in terms of the *mean absolute percent error (MAPE)*; however, the costs of the error are too high to be acceptable. Electric utilities require a much tighter operations load forecast performance. For example, Electric Reliability Council of Texas (ERCOT) requires that the monthly averaged MAPE for day ahead load forecasts be less than 4%.

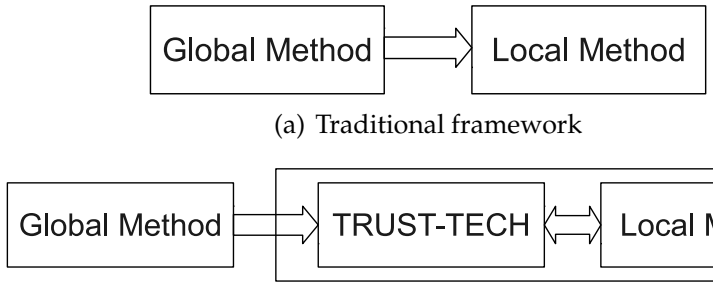
A wide variety of models have been proposed for load forecasting in the last three decades owing to its importance. Time series analysis based approaches had been tried with varying degrees of success, such as autoregressive models (ARMA [82] and ARMAX [179]), Kalman filtering [84, 137], optimization techniques [187], non-parametric regression [28], and linear regression [75, 158]. These models are basically linear analysis and they lack the capability to model the underlying complex and nonlinear relationship between the load and the factors influencing the load. Therefore, new forecasting models, mainly artificial intelligence techniques, have been recently introduced to solving the load forecasting problem. Among these techniques, expert systems [80, 97, 140], fuzzy inference [127], fuzzy-neural models [47, 96, 97, 112, 151] and support vector machines [29, 55] have been tried out. However, the models that have received the largest share of attention are undoubtedly artificial neural networks (ANNs) [9, 44, 106, 117, 124, 135, 136, 166, 185]. It seems that ANN-based forecasting systems have been well accepted in practice and are used by many util-

ities [94, 95]. Nevertheless, reviews of ANN-based forecasting systems have concluded that they are promising but much work still needs to be done before they are accepted as established forecasting techniques [16, 188].

This chapter focuses on short-term load forecasting, in particular on forecasting a day-ahead load profile, which means that 24 load forecasts are computed for each day of the week. To develop such a forecasting system, we employ the ELITE methodology presented in Chapter 6 to construct two neural network ensembles to forecast the full load profile and the differential load profile, respectively. Outputs of these ensembles are then combined to get the desired forecast of the load profile in the next day. In order to better initialize the member neural networks involved in the ensembles, a TRUST-TECH based hybrid framework is first introduced. Under this framework, a TRUST-TECH based evolutionary programming (TT-EP) is developed to acquire neural network parameters in promising areas of the search space. The constructed forecaster has been tested on utility production data and the numerical results show that a promising performance with the MAPE being 1.88% is achieved, which outperforms other four popular models with a noticeable performance margin ranging from 55.8% to 139.36%.

7.2 TRUST-TECH Based Hybrid Framework

Existing optimization methods can be categorized into *local* and *global* methods. Local optimization methods, such as the *back-propagation (BP) algorithm* [88], *scaled conjugate gradient (SCG) method [98]* and *Levenberg-Marquardt (LM) algorithm* [70], have received significant attention. Although these local methods



(a) Traditional framework

(b) Proposed framework

Figure 7.1: Comparison between different frameworks. Including TRUST-TECH in the proposed framework results in a better cooperation between local and global methods.

are relatively effective in terms of computational requirements, they can only attain a local optimal solution close to the initial condition. This local optimal solution is usually not the best in the search space. On the other hand, global methods such as *simulated annealing (SA)* [118], *Tabu search* [184], *ant colony optimization* [156], *evolutionary programming (EP)* [63] and *particle swarm optimization (PSO)* [90] can identify promising regions of the search space, but they are stochastic in nature and computationally expensive. Combined local and global methods are popular in practical applications (Fig. 7.1(a)) [24, 49, 105]. However, traditional approaches can still find local optimal solutions close to the initial points provided by the global method. It is likely that better local optimal solutions can be found in the vicinity of the obtained local optimal solutions (Fig. 7.2(a)).

This section presents a TRUST-TECH based hybrid framework, where existing local and global methods can be bridged in a better manner. Taking advantage of deterministic and systematic features TRUST-TECH, an effective platform is introduced to cooperate with existing local and global methods. This cooperation starts with a global method to obtain promising regions; then, mul-

tiple local optimal solutions are computed by efficiently examining neighbouring subspaces using TRUST-TECH with robust and fast local methods. In this way, high-quality local optimal solutions or the global optimal solution can be secured (Fig. 7.2(b)).

7.2.1 The hybrid framework

The TRUST-TECH methodology computes multiple local optimal solutions in a tier-by-tier manner. If the initial local optimal solution is far away from the global optimal solution, however, the number of search tiers can be large. On the other hand, although global methods are notoriously inefficient in finding good approximates to the global optimal solution, they are good at finding promising regions. Effectively incorporating the strength of existing methods and that of TRUST-TECH motivates us to develop the proposed framework.

Structure of the TRUST-TECH based hybrid framework is shown in Fig. 7.3. Specifically, two stages are included in this framework. The first stage is to perform a global search using a global method and the target is to obtain promising regions. Since it is not used solely to compute the global optimal solution, the number of iterations required by the global method can be significantly reduced. As a result, scalability of the global method can be improved. In the second stage, promising regions and their neighbouring subspaces are effectively examined using TRUST-TECH with robust and fast local methods. Multiple local optimal solutions in the promising regions are computed in a tier-by-tier manner. From these local optimal solutions, high-quality local optimal solutions or the global optimum can be obtained.

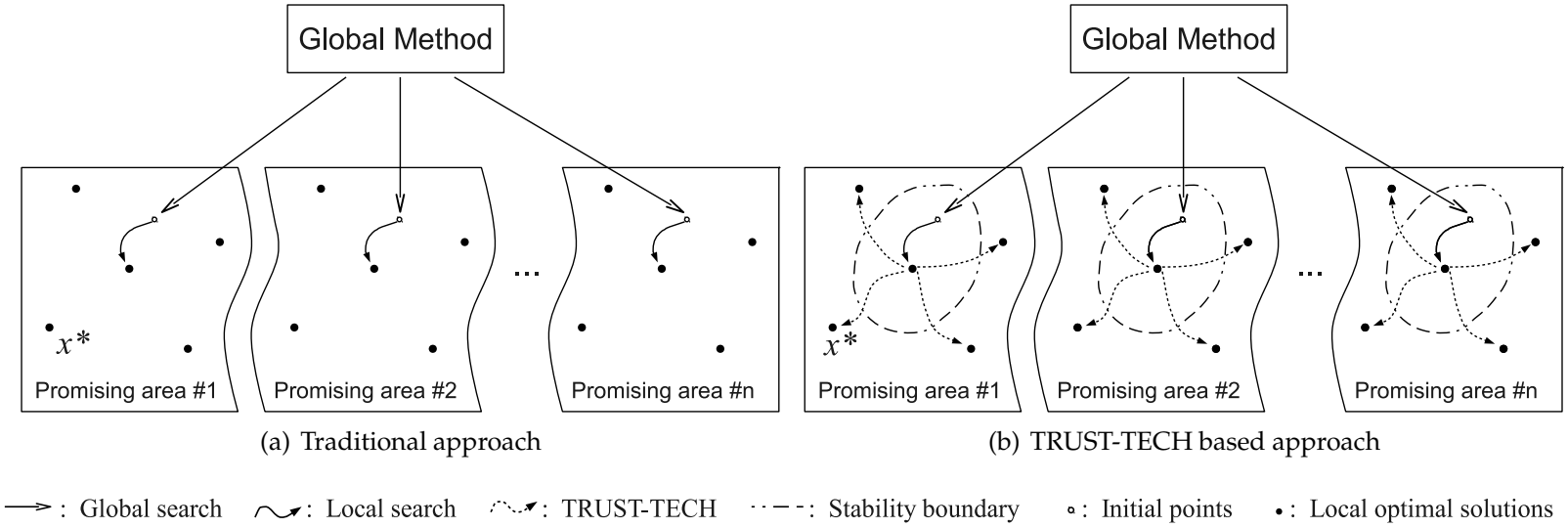


Figure 7.2: The traditional global + local approach lacks the capability for fine-tuning, but better solutions can usually be found in the vicinity, as illustrated in (a) where x^* is the global minimum. Empowered with the stability region based dynamical phase, the TRUST-TECH based approach possesses the capability to access neighbourhood local optimal solutions via a tier-by-tier search, thus those better solutions will not be missed, as illustrated in (b).

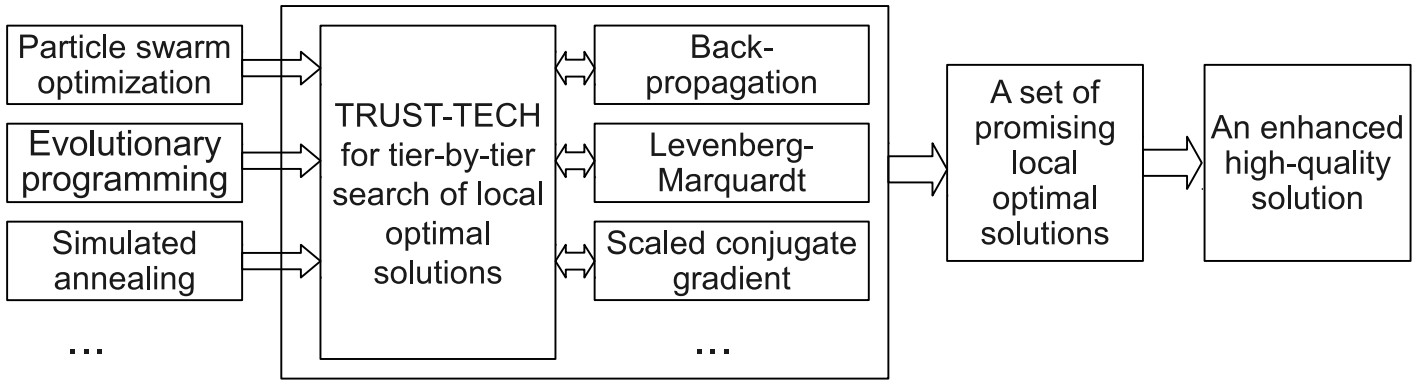


Figure 7.3: Detailed structure of the TRUST-TECH based hybrid framework. In this framework, TRUST-TECH provides an effective platform to better cooperate existing local and global methods.

Features of the proposed framework include: 1) it can provide an effective platform for cooperating local and global methods by enabling traditional methods to zoom-in promising regions to find high-quality local optimal solutions; 2) it allows a flexibility of choosing different local and global methods to tackle different problems effectively.

7.2.2 Evolutionary programming

Research efforts on developing computational models have been rapidly growing in recent decades. Amongst the several ways of developing effective computational models, evolutionary computation has become very popular. The evolutionary computational models [60] use well-studied computational models of evolutionary processes as key elements. There are a variety of such models that have been proposed and studied which we will refer to as *evolutionary algorithms (EAs)*. In simple terms, they simulate the process of evolution of individual components via processes of selection and reproduction. These processes depend on the fitness of individuals as defined by an environment. Evolutionary algorithms maintain a population of individuals that evolve according to rules of selection and other genetic operators, such as recombination and mutation. Each individual in the population receives a measure of its fitness in the environment. Of all these operations mentioned above, selection is the main one that can exploit the available fitness information and mainly considers those individuals with high fitness values. Recombination and mutation perturb these individuals, providing general search strategies and heuristics for exploring the solution space.

These evolutionary algorithms are sufficiently complex to provide robust and powerful adaptive search mechanisms, especially when the number of local optimal solutions grow exponentially with the problem dimension. Meanwhile, since no gradient information is required, this category of methods is also generally easy to implement. As a consequence, evolutionary algorithms have been widely used to tackle different tasks in learning neural networks [5, 63, 67, 134]. An in-depth review of different combinations between neural networks and evolutionary algorithms can be found in [181].

Among the category of evolutionary algorithms, the *evolutionary programming (EP)* and *evolutionary strategy (ES)* are particularly well-suited for treating continuous optimization. Unlike *genetic algorithms (GA)*, the primary search operator in EP and ES is mutation. One of the major advantage of using mutation-based evolutionary algorithms is that they can reduce the negative impact brought by the permutation problem [181]. Therefore, the evolutionary process can be more efficient in these two algorithms.

In this chapter, the evolutionary programming will be the global method considered in realizing the TRUST-TECH based hybrid framework. The generic algorithm for an EP algorithm includes the following steps for neural network training:

Step 1) Initialization: Generate an initial population of P individuals with random values, and set the iteration counter to be $k = 1$. Each individual is a pair of real valued vectors $(\mathbf{x}_i, \mathbf{v}_i)$, $\forall i \in \{1, \cdots, P\}$, where $\mathbf{x}_i \in R^n$ are vectors of network weights and biases, and $\mathbf{v}_i \in R^n$ are vectors of variance for Gaussian mutations. Therefore, each individual corresponds to a neural network with the same structure.

Step 2) Generating the offspring: At the t-th iteration, each individual in the population $(\mathbf{x}_i, \mathbf{v}_i), \forall i \in \{1, \cdots, P\}$, produces a single offspring $(\mathbf{x}'_i, \mathbf{v}'_i)$ via the following computations:

$$\begin{aligned} v'_i(j) &= v_i(j) \cdot \exp(\tau_1 N(0,1) + \tau_2 N^j(0,1)) \\ x'_i(j) &= x_i(j) + v'_i(j) \cdot N^j(0,1) \end{aligned}, \tag{7.1}$$

where, $j = 1, \cdots, n$, $x_i(j)$, $x'_i(j)$, $v_i(j)$ and $v'_i(j)$ denote the j-th components of the vectors $\mathbf{x}_i$, $\mathbf{x}'_i$, $\mathbf{v}_i$ and $\mathbf{v}'_i$, respectively. $N(0,1)$ denotes a normally distributed one-dimensional random number with mean 0.0 and variance 1.0. $N^j(0,1)$ indicates that the random number is generated anew for each value of j. τ_1 and τ_2 are constants, which are commonly set to be $(\sqrt{2n})^{-1}$ and $\left(\sqrt{2\sqrt{n}}\right)^{-1}$, respectively.

Step 3) Fitness evaluation: Determine the fitness of every individual in the population, including all parents and their offspring. In other words, the training data is applied to each neural network corresponding to the individuals and the training performance is evaluated.

Step 4) Updating the population: Conduct a pairwise comparison over the union of parents $(\mathbf{x}_i, \mathbf{v}_i)$ and their offspring $(\mathbf{x}'_i, \mathbf{v}'_i)$, $\forall i \in \{1, \cdots, P\}$. For each individual, q opponents are chosen uniformly at random from all parents and their offspring. In each comparison, if the individual's fitness is no smaller than that of the chosen opponents, it receives a *win*. Select P individuals out of $(\mathbf{x}_i, \mathbf{v}_i)$ and $(\mathbf{x}'_i, \mathbf{v}'_i)$, $\forall i \in \{1, \cdots, P\}$, that have the most wins to form the next generation.

Step 5) Checking the stopping criterion: Stop the whole evolutionary process if the stopping criterion is satisfied; otherwise, increment the iteration counter to $k = k + 1$ and go to *Step 2)* to start a new evolutionary iteration.

7.2.3 The TT-EP method

Generally, a large number of generations are required in order for an EP with a reasonable population size to converge to good solutions. Therefore, training neural networks using evolutionary programming solely can be very time consuming and not effective. However, as a global method, one advantage of EP is its ability to find promising regions containing high quality solutions. Using EP as a good initializer to identify promising regions in the high-dimensional search space, TRUST-TECH can work with local optimizers to efficiently explore these regions for high-quality solutions. In this manner, EP can be applied only for a significantly small number of iterations and it is not required to converge to the solutions.

Therefore, following the proposed hybrid framework, a TRUST-TECH enhanced EP (TT-EP) training method can be developed, which is described in detail as follows.

The TRUST-TECH enhanced EP training method

Input: the population size P, τ_1 and τ_2 for EP, and the maximal number of TRUST-TECH search tiers T_m.

Output: an optimal weight vector $\mathbf{x}^*$ and a set $\mathbf{X}_s$ of local optimal solutions.

Initialization: initialize the EP population and set $\mathbf{X}_s = \emptyset$.

Algorithm:

1) Carry out the standard EP algorithm on the population.

2) Choose I best individuals $\{\mathbf{x}_1, \cdots, \mathbf{x}_I\}$ from the population.

3) **for** $i = 1 : I$, **do**

   - Using $\mathbf{x}_i$ as the initial point, retrain the neural network by a local method, such as BP, until convergence and get the local optimum $\mathbf{x}_s$.
   - Using $\mathbf{x}_s$ as the initial local optimum, carry out multi-tier TRUST-TECH search to get a set of local optimal solutions $\{\mathbf{x}_{s1}, \cdots, \mathbf{x}_{sk}\}$.
   - $\mathbf{X}_s = \mathbf{X}_s \cup \{\mathbf{x}_{s1}, \cdots, \mathbf{x}_{sk}\}$

4) Determine the optimal weight vector:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}}\{f(\mathbf{x})|\mathbf{x} \in \mathbf{X}_s\}$$

7.3 The ELITE-STLF System

7.3.1 System structure

In this section, a system for short-term load forecasting, called ELITE-STLF, is developed using the ELITE method presented in Chapter 6 and the TT-EP method described in Section 7.2. The structure of the ELITE-STLF system is shown in Fig. 7.4. ELITE-STLF has a similar structure as that of ANNSTLF (generation 3) which is reported in [95], since both systems are composed of two sub-forecasters. The first sub-forecaster predicts the 24-hour load curve, and the second one predicts the change of the load curve with respect to the load curve of the previous day. The ELITE-STLF system differentiates itself from the

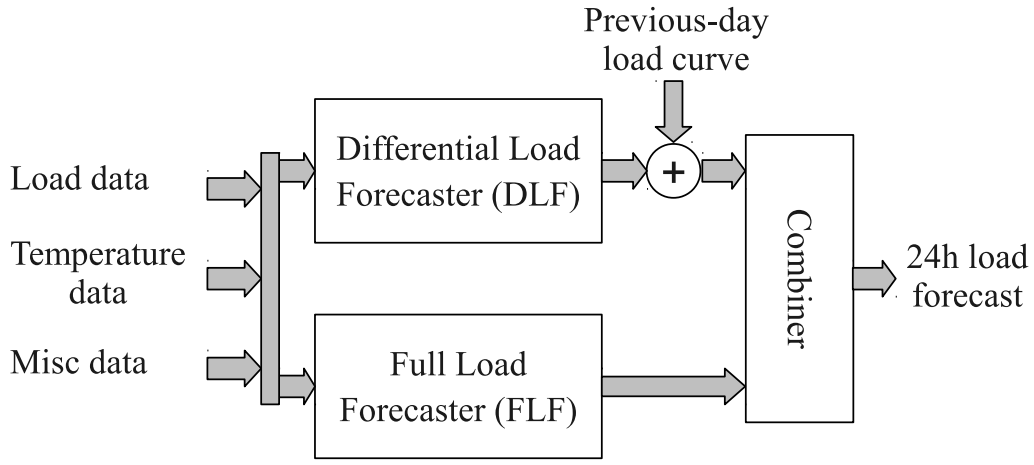


Figure 7.4: This figure depicts the ELITE-STLF system for short-term load forecasting, where two forecasters are constructed and their outputs are combined. One forecaster predicts the next-day 24-hour load curve, while the other forecasts the 24-hour next-day change of the load with respect to the previous day.

ANNSTLF system in that each sub-forecaster is an ensemble of neural networks constructed using the ELITE method.

As is shown in Fig. 7.4, the ELITE-STLF system consists of three modules, including two ANN ensemble load forecasters and a linear combiner. Both load forecasters receive the same set of inputs and produce a load forecast for the same day, but they utilize different strategies to implement the forecast. The function of the combination module is to mix the two forecasts to generate the final forecast with improved accuracy.

The difference between the two sub-forecaster is in their outputs. The first ensemble forecaster is trained to predict the regular (full) load of the next day, that is, the 24 outputs are the forecasts of the hourly loads of the next day. This sub-forecaster is referred to as the *Full Load Forecaster (FLF)*. On the other hand, the second ensemble forecaster predicts the difference in hourly loads from the

previous day to the forecasting day. This forecaster is named *Differential Load Forecaster (DLF)*. Hence, the outputs of the two sub-forecasters are (the subscript k denotes the forecasting day):

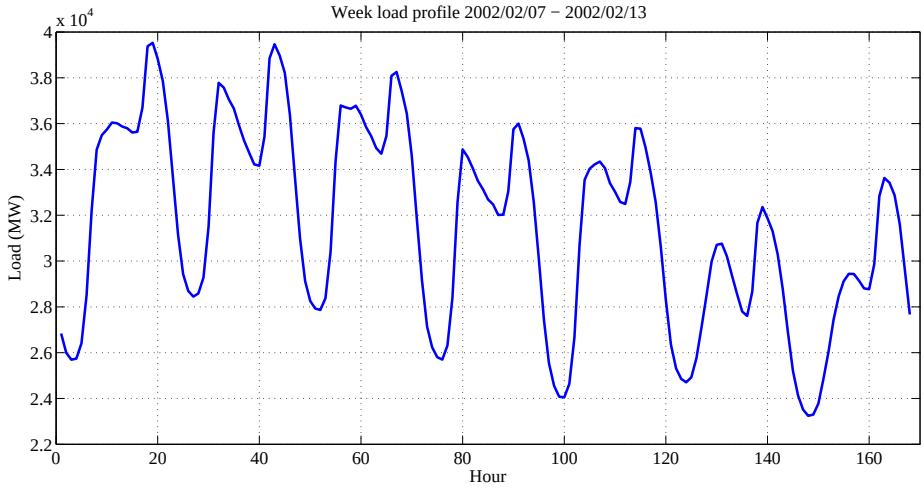
- FLF: $\hat{L}_k^F(1), \hat{L}_k^F(2), \cdots, \hat{L}_k^F(24)$.
- DLF: $\Delta\hat{L}_k^D(1), \Delta\hat{L}_k^D(2), \cdots, \Delta\hat{L}_k^D(24)$, where $\Delta\hat{L}_k^D(i) = \hat{L}_k^D(i) - \hat{L}_{k-1}^D(i)$ is the difference of the i-th hour load from day $k-1$ to the forecasting day k.

To get full load forecasts from the DLF, the outputs of the DLF module are added to the loads of previous day, that is,

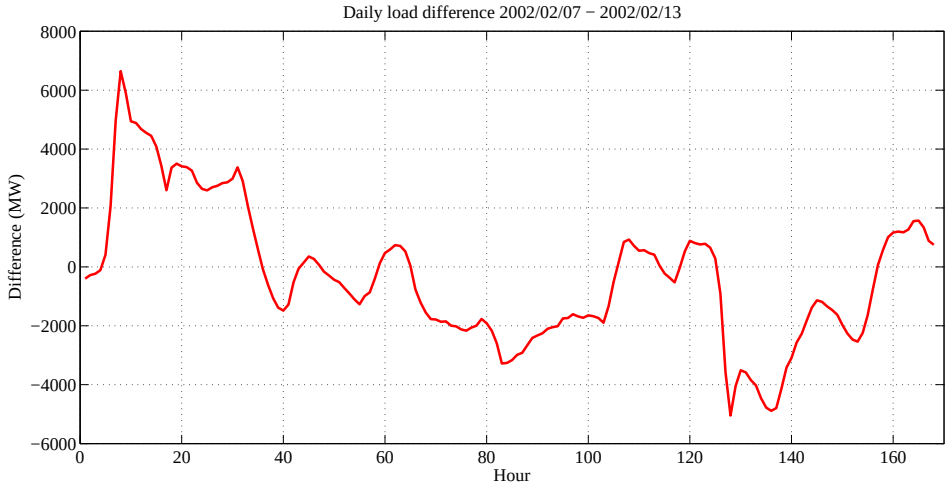
$$\hat{L}_k^D(i) \;=\; \Delta\hat{L}_k^D(i) + \hat{L}_{k-1}^D(i) \tag{7.2}$$

The two sub-forecasters complement each other because the FLF emphasizes regular load patterns whereas the DLF puts stronger emphasis on previous day's load pattern. Combining these two separate forecasts will result in improved performance. This is especially true for cases of sudden load changes caused by weather fronts. The FLF has a tendency to respond slowly to rapid changes in loads. On the other hand, since the DLF takes previous day's loads as the basis and predicts the differences in loads, it has a faster response to a changing situation. Patterns of the full and differential loads are illustrated in Fig. 7.5 for a week period from January 07, 2002 to January 13, 2002. It can be observed from Fig. 7.5(a) that the full load profile is much smoother and shows a daily periodic pattern. In contrast, the differential load profile between two consecutive days is irregular and shows no periodicity, as shown in Fig. 7.5(b).

Taking advantage of the complimentary performance of the two sub-forecasters, the system forecast for each hour is obtained by a linear combination



(a) Weekly load profile



(b) Load difference profile

Figure 7.5: Weekly load and difference profiles. The first figure shows the weekly load profile during 2002/01/07 to 2002/01/13. The second figure shows the hour-wise load change from previous day.

of the FLF and DLF outputs as:

$$\hat{L}_k(i) = \alpha \hat{L}_k^F(i) + \beta \hat{L}_k^D(i) \tag{7.3}$$

The coefficients α and β can be constant or can be adaptively learned to maximize the system performance. In the current implementation of ELITE-STLF,

the values of α and β are constant and set to $\alpha = \beta = 0.5$. In other words, the system forecast is the averaged output of the FLF and DLF outputs.

7.3.2 Constructing sub-forecasters

Both FLF and DLF have the same topology and are constructed through the same procedure. The procedure to construct an ensemble forecaster is shown in 7.6. Two stages are involved in learning a sub-forecaster, that is, the TT-EP stage to compute promising points and the ELITE stage to build the ensemble.

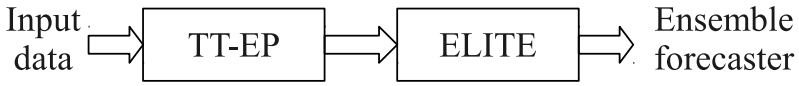


Figure 7.6: The process to construct sub-forecasters. There are two stages, TT-EP and ELITE, involved in constructing each of the two sub-forecasters of ELITE-STLF.

The first stage is TT-EP. The purpose of this stage is to take advantage of global search methods to find promising points in the search space. The TT-EP is performed on the neural network weights. Therefore, each individual in the EP population corresponds to a neural network with the same structure (number of inputs, outputs, layers and hidden nodes) but different weight values.

The main reason for choosing EP, instead of other evolutionary algorithms such as PSO, is that its final population tends not to converge to the same best solution. In other words, the solution points obtained by EP tend to be distributed in different promising areas. This will contribute to the member diversity for ensemble.

The second stage is ELITE. It is known that ensemble offers an effective way

to alleviate the burden of tuning the parameters of a single ANN and usually results in improved generalization capability. Using the promising points produced by TT-EP, ELITE is performed to obtain optimally prune member networks. Then optimal combination weights are calculated and used to construct the ensemble forecaster. In this way, high-quality ensembles can be obtained to serve as accurate FLF and DLF forecasters.

7.4 Numerical Study

In this section, the proposed ELITE-STLF method will be applied for day-ahead load forecasting on a set of practical utility data. The testing result reveals that ELITE-STLF can achieve a performance with MAPE being 1.88%, which shows that ELITE-STLF is a promising forecaster for practical use. The experimental set-up and results are described in the remaining part.

7.4.1 The dataset

The load and temperature data provided by PJM is used to test the proposed ELITE-STLF method. The data ranges from 00:00, January 1, 2000 to 23:00, June 19, 2002, covering two and half years. Measurement is taken at each clock, hence there are 24 load and temperature data points for each day. Data for the first two years (from 00:00, January 1, 2000 to 23:00, December 31, 2001) is used for training, and the data for the remaining half year (from 00:00, January 2002 to 23:00, June 19, 2002) is used for testing.

Let the k-th day be the forecasting day. The data vector input to the ELITE-

STLF system, which is of 108 dimensions, is organized as follows:

- *Load data:* The load data consists of the 24-hour load data of the previous day ($(k-1)$-th day), the day before the previous day ($(k-2)$-th day), and the same day in the last week ($(k-7)$-th day). Hence, the load data is 72-dimensional.
- *Temperature data:* The temperature data consists of the minimal, maximal and average temperature of the $(k-1)$-, $(k-2)$- and $(k-7)$-th days, and the forecasting 24-hour temperature of the k-th day. The temperature data is of 33 dimensions.
- *Misc data:* The misc data consists of the weekday index. This weekday index is encoded in 3 bits, ranging from 001 to 111. The main consideration to employ this scheme is that the probability of 0 and 1 for each input is roughly balanced in this way. In contrast, for the 7-bit scheme used by the existing methods, the presence of 0 and 1 in each bit is severely unbalanced.
- *Holidays:* In the current implementation, holidays are treated as weekends. Weekdays before and after non-weekend holidays will be treated as Friday and Monday, accordingly.

The output of the forecaster is a 24-dimension vector corresponding to the load profile in the forecasting day.

7.4.2 Experimental setup

The ELITE-STLF forecaster has been implemented using MATLAB. Two layer feed-forward neural networks are used as the member network. The *hyperbolic tangent function* is used as the transfer function in the hidden layer and the *linear function* for the output layer. The size of the hidden layer is 36. In the evolutionary programming procedure for computing promising points, the population size is 50 and the evolution is performed for 1000 generations. From the final population, 20 best individuals are selected for feature selection and optimal combination in ELITE. The *scaled conjugate gradient method* is used as the local optimizer for weight training.

As preprocessing, every dataset (both the training and testing samples) is linearly scaled (item-wise) via the following transformation before being used to construct and test the forecaster:

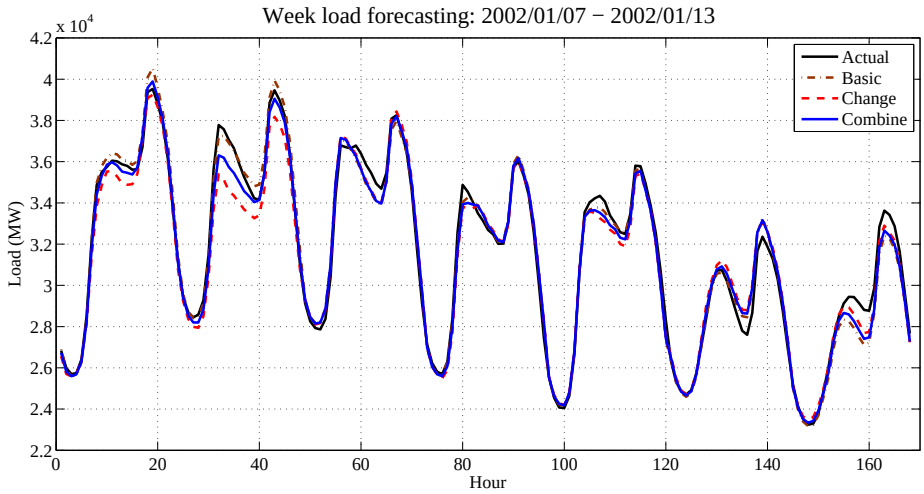
$$\tilde{\mathbf{x}} = 2 \times \frac{\mathbf{x} - \mathbf{min}_{tr}}{\mathbf{max}_{tr} - \mathbf{min}_{tr}} - 1.0 \tag{7.4}$$

where, $\mathbf{min}_{tr}$ and $\mathbf{max}_{tr}$ are vectors of the minimum and the maximum values for the training samples, respectively. In other words, the dynamic range of the training data (both input and output) is linearly scaled to $[-1.0, 1.0]$.

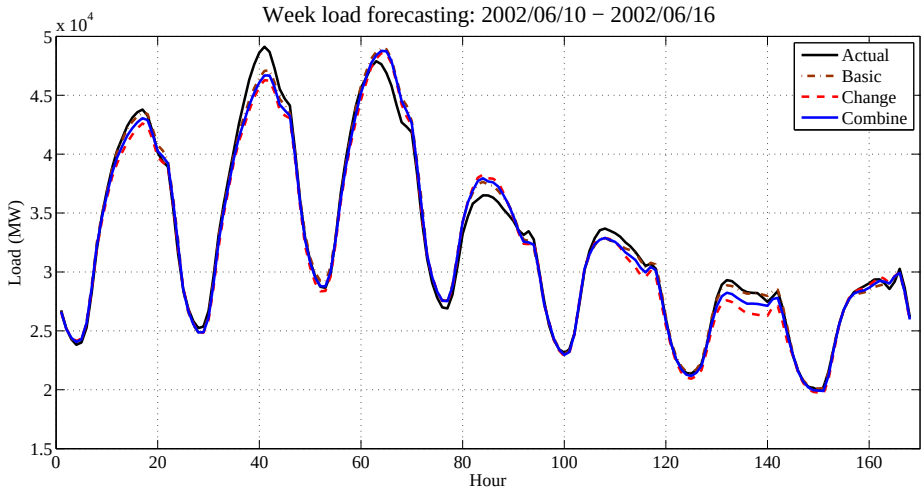
7.4.3 Results

The forecasting performance is represented by the *mean absolute percent error (MAPE)* which is evaluated as follows:

$$MAPE = \frac{|\text{Forecasted load} - \text{Actual load}|}{\text{Actual load}} \times 100\% \tag{7.5}$$



(a) Week 2002/01/07 - 2002/01/13



(b) Week 2002/06/10 - 2002/06/16

Figure 7.7: The forecasting results for different weeks in two seasons. It is obvious that the forecasting error happens mostly at the peak load area on the load profile.

The actual and forecast load profiles for two weeks (the week of January 7, 2002 to January 13, 2002 and the week of June 10, 2002 to June 16, 2002) in two seasons are shown in Fig. 7.7. A close match between the forecasts and the actual loads can be observed, which shows the effectiveness of the ELITE-STLF system.

Table 7.1: The ELITE-STLF system forecasting performance summary. Combining the FLF and DLF results in lower MAPE than both sub-forecasters.

	Full load forecaster	Differential load forecaster	System forecasting
MAPE	1.95%	2.11%	1.88%

Table 7.2: The forecasting performance for daily special loads.

	Daily peak forecasting	Daily valley forecasting	Daily mean forecasting
MAPE	2.21%	1.08%	1.60%

The forecasting MAPE of the whole system and that of the full load forecaster and the differential load forecaster are summarized in Table 7.1. It is shown that the MAPE by the full load forecaster alone is 1.95% and the MAPE by the differential load forecaster is 2.11%. By combining the outputs of these two sub-forecasters, the system-wise forecasting MAPE is reduced to 1.88%. In other words, the combined forecaster outperforms both sub-forecasters, with improvements being 3.59% and 10.90%, respectively.

The forecasting performance is also evaluated for daily special loads. Specifically, the MAPEs associated with the daily peak, valley and averaged loads are calculated and shown in Table 7.2. From this table, we can see that the largest forecasting error happens at the peak load, with the MAPE being 2.21%. This error is about 17.55% higher above the system MAPE. It can also be seen that the forecaster performs quite well in following the daily valley and mean loads. These observations can also be verified by inspecting the weekly load curves

Table 7.3: The averaged forecasting performance for individual hours.

Hour	00:00	01:00	02:00	03:00	04:00	05:00
MAPE	0.69%	0.78%	0.94%	1.05%	1.19%	1.38%
Hour	06:00	07:00	08:00	09:00	10:00	11:00
MAPE	1.75%	1.87%	1.77%	1.81%	1.91%	2.02%
Hour	12:00	13:00	14:00	15:00	16:00	17:00
MAPE	2.14%	2.27%	2.43%	2.49%	2.52%	2.60%
Hour	18:00	19:00	20:00	21:00	22:00	23:00
MAPE	2.52%	2.40%	2.17%	2.08%	2.08%	2.19%

Table 7.4: The averaged forecasting error for individual weekdays.

Weekday	MON	TUE	WED	THU	FRI	SAT	SUN
MAPE	2.05%	1.69%	1.67%	1.62%	1.60%	1.94%	2.59%

shown in Fig. 7.7 and the hour-wise performance summarized in Table 7.3.

Table 7.4 summarizes the forecasting performance for individual weekdays. Based on this table, it can be seen that the forecaster performs better on normal weekdays (Tuesday through Friday) than on weekends. Performance on Monday is negatively affected by loads on Sunday. Such difference happens because normal weekdays are mostly workdays and possess more regularity in terms of electricity consumption. Further study is needed to improve the forecasting accuracy for special days, including weekends and holidays.

The averaged forecasting performance in individual months is also evaluated, which is summarized in Table 7.5. This table suggests that the forecasting

Table 7.5: The averaged forecasting performance in individual months.

Month	January	February	March	April	May	June
MAPE	1.47%	1.52%	2.15%	1.85%	2.33%	1.93%

performance tends to degrade from month to month. The best performance is seen in January, 2002, with the MAPE being 1.47%. This is due to that the forecaster is fixed once it has been constructed based on the historical data. Apparently, the forecaster favours the time period close to the history data which has been memorized by it. In order to prevent the performance from degrading, the forecaster should be retrained every other week or month using the up-to-date load and weather data.

7.4.4 Comparison with other methods

Four existing methods have also been implemented and applied to the same dataset. These methods include:

1) The time series *auto regressive moving average (ARMA)* model [82];

2) The time series *auto-regressive moving average with exogenous inputs (ARMAX)* model [179];

3) The *nonlinear autoregressive dynamic neural network with exogenous inputs (NARX NN)* method;

4) *Support vector regression (SVR)* using LibSVM [26].

Table 7.6: Comparison of the performance with other methods.

Method	ELITE	ARMA	ARMAX	NARX NN	SVR
MAPE	1.88%	2.93%	4.0%	4.50%	3.65%

Performance of these methods are evaluated and compared with ELITE-STLF. The MAPEs achieved by these methods are summarized in Table 7.6. The following observations can be obtained based on the comparison:

- The two autoregressive methods, ARMAX and NARX, have the worst performance, with MAPEs being 4.0% and 4.5%, respectively. It is because these models are basically linear analysis and they lack the capability to model the underlying complex and nonlinear relationship between the load the factors influencing the load.
- Compared with ARMAX and NARX, the SVR method achieves an improved MAPE of 3.65%. This is because SVR is able to model nonlinear relationships between the input and the output.
- The ELITE-STLF method outperforms all other methods on the dataset, with a noticeable performance margin ranging from 55.8% to 139.36%. This reveals that the underlying complicated nonlinear relationship between the load the factors influencing the load can be well modeled in the proposed method.

7.5 Summary

In this chapter, the ELITE method is applied to construct a high-performance short-term load forecaster, called ELITE-STLF. Following the TRUST-TECH based hybrid framework, the evolutionary programming is used in ELITE-STLF to find promising points in the search space. The forecaster is tested using utility production data and the result shows that a good performance with the MAPE being 1.88% is achieved. This implies that ELITE-STLF is a very promising and competitive option in practical implementations.

Further studies in the following aspects may prove fruitful:

1) Peak load correction. The results have indicated that most of the forecasting error happens at the peak are on the daily load profile. In fact, daily peak loads play the most important role for utilities in their planning of power generations. In other word, further improvement is necessary and beneficial to enhance the forecasting accuracy for the daily peak loads.

2) Output combination. The current ELITE-STLF gets the system output by averaging the outputs of FLF and DLF. Introducing another module dedicated to learn the combination weights could be beneficial. Specifically, this extra module will receive the same input as other sub-forecasts. The output is also 24 dimension, corresponding to 24 combination weights. The output values for training can be calculated by mean square error minimization or by L_1 norm minimization.

3) Adaptive learning. The current implementation of ELITE-STLF is trained off-line. In other words, the structure and parameters are fixed once the fore-

caster has been constructed using the historical data. An adaptive and online learning mechanism should be introduced into ELITE-STLF such that new data can be involved in the learning process constantly. In this way, the long-term performance of the forecaster can be further improved.

4) Wind forecasting. Wind power is anticipated to increase steadily as the demand raised for clean and renewable energy. An accurate wind forecasting is indispensable once the wind power becomes a non-negligible portion to the total generation. As a new application area, the ELITE method could also be a promising technique for wind or wind power forecasting. Such wind forecaster could also be integrated into ELITE-STLF to further improve the load forecasting accuracy.

CHAPTER 8

CONCLUSION AND FUTURE WORK

This chapter concludes our discussion and highlights the most important contributions of this thesis. It also discusses future research directions that could further extend TRUST-TECH's capability to attack a broader range of optimization problems.

8.1 Conclusion

This work extends the TRUST-TECH methodology by incorporating new analytical results, developing new solution methods and solving new problems in practical applications. We have demonstrated the applicability and effectiveness of these methods for practical and high-dimensional nonlinear optimization problems. The central idea behind these methods is to transform the original optimization problem into a dynamical system with certain properties and to obtain more useful information about the nonlinear surface via analysing dynamical and topological properties of the dynamical system.

In Chapter 2, we studied relationships between a gradient dynamical system and its associated quasi-gradient variant and revealed the invariance of partial stability region in the quasi-gradient system. Via these analyses, we answered the question regarding invariant convergence for a special class of numerical operations whose dynamical behaviour can be characterized by a quasi-gradient dynamic system. Based on these analytical results, we also developed algorithms for checking and preserving invariant convergence of the trajectory starting from a given point in the quasi-gradient system.

This work also developed new solution methods to enhance TRUST-TECH's capability of solving constrained nonlinear optimization problems. In Chapter 3, we first developed TRUST-TECH based methods for feasibility computation and restoration. Compared with unconstrained problems, the existence of nonlinear constraints in constrained problems make the optimization task much more complicated. Analysing feasibility of the problem and attaining a feasible solution is usually the first step in solving a constrained optimization problem. Indeed, a unified framework based on the TRUST-TECH methodology has been proposed for analysing feasibility and infeasibility of nonlinear problems. These methods were applied to power system applications, including power flow computation and feasibility restoration for infeasible OPF problems, with promising results on large systems.

Local methods are usually deterministic and computationally efficient in attaining a local optimal solution. Cooperation of TRUST-TECH with local methods will result in effective methods for global optimization of constraint nonlinear programs. Following this spirit, Chapter 4 developed the *TRUST-TECH based interior point method (TT-IPM)*. In Chapter 4, the *reduced projected gradient method* was also developed to solve nonlinear optimization problems with a two-phase strategy. The TT-IPM method was used to solve *mixed-integer nonlinear programs (MINLPs)* in Chapter 5 with interesting results.

In Chapter 6, we developed the *ensemble of optimal, input-pruned neural networks using TRUST-TECH (ELITE)* method for constructing high-quality neural network ensembles. Optimization problems involved in ensemble, including optimal training, diversity improving and optimal combination, are effectively solved using TRUST-TECH based methods. The ELITE method was then ap-

plied to build a short-term load forecaster in Chapter 7. The constructed forecaster, named ELITE-STLF, outperforms several existing methods when applied on a utility provided production dataset.

8.2 Future Work

As future efforts, the TRUST-TECH methodology could be generalized in the following aspects with even far-reaching influence in the optimization field and be empowered with the ability to attack a broader range of optimization problems.

Firstly, the TRUST-TECH methodology could be applied to solve the class of multi-objective optimization problems. The algorithms developed in this work are mainly focused on optimization problems with single objective. In fact, many optimization problems in various fields require to optimize multiple objectives at the same time. Solving these optimization problems is the task of multi-objective optimization. In the mono-objective optimization scenario, we generally can obtain a single (local) optimal solution since the objective space is naturally ordered. For *multi-objective programs (MOPs)*, however, the objective space becomes multi-dimensional and there is no natural ordering because it is only partially ordered. Hence, we generally can get a set of solutions (the Pareto optimal set) to an MOP such that each solution is not worse (inferior) than any other solutions. In addition, for nonlinear MOPs, there could exist multiple local Pareto optimal sets. TRUST-TECH could be a promising technique used to develop methods to compute multiple solutions lying on a Pareto optimal set and to find multiple Pareto optimal sets for nonlinear MOPs.

Secondly, the spirit rooted in the TRUST-TECH methodology could also be generalized into infinite- dimensional spaces and be used to solve optimization problems defined in these spaces. It needs to be noticed that analyses and methods related to the current TRUST-TECH methodology concern exclusively about optimization problems defined in finite-dimensional vector spaces and characterized by ordinary calculus. In contrast, variational optimization, or calculus of variations, is a field of mathematics that deals with functionals, as opposed to ordinary calculus which deals with functions. These functionals can, for example, be formed as integrals involving an unknown function and its derivatives. The interest is in extremal functions which make the functional attain a maximum or minimum value or stationary functions where the rate of change of the functional is precisely zero. Many applications, such as image segmentation and restoration and optimal control problems, can be modelled as variational optimization problems. In order to tackle this class of problems, the theoretical foundation of the TRUST-TECH methodology, that is, characterization of stability regions of nonlinear dynamical systems needs first to be generalized to more general spaces where variational problems can be defined. One type of such spaces of interest is the class of Banach spaces. Then, based on such generalization, TRUST-TECH based methods could be developed in these spaces to solve variational optimization problems.

In concluding the entire work, the spirit in the TRUST-TECH methodology of locating multiple local optimal solutions with the aid of suitable nonlinear dynamical systems can be generalized to a much broader range of nonlinear optimization models and be used to enhance the capability of a much wider class of optimization algorithms.

BIBLIOGRAPHY

[1] I. Adler, N. Karmarkar, M. G. C. Resende, and G. Veiga. An implementation of karmarkar's algorithm for linear programming. *Math. Program.*, 44(1-3):297–335, 1989.

[2] E. Amaldi, M. Bruglieri, and G. Casale. A two-phase relaxtion-based heuristic for the maximum feasible subsystem problem. *Comput. Oper. Res.*, 35(5):1465–1482, 2008.

[3] P. Amaral and P. Barahona. Connections between the total least squares and the correction of an infeasible system of linear inequalities. *Linear Algebra Appl.*, 395:191–210, 2005.

[4] S. Amato, B. Apolloni, et al. Simulated annealing approach in backpropagation. *Neurocomputing*, 3(5):207–220, 1991.

[5] P. J. Angeline, G. M. Saunders, and J. B. Pollack. An evolutionary algorithm that constructs recurrent neural networks. *IEEE Trans. Neural Netw.*, 5(1):54–65, 1994.

[6] A. Asuncion and D. J. Newman. UCI machine learning repository, 2007. [Online]. Available: `http://archive.ics.uci.edu/ml/`.

[7] E. H. Barakat and J. M. Al-Qasem. Methodology for weekly load forecasting. *IEEE Trans. Power Syst.*, 13(4):1548–1555, 1998.

[8] M. E. Baran and F. F. Wu. Optimal capacitor placement on radial distribution systems. *IEEE Trans. Power Del.*, 4(1):725–734, 1989.

[9] T. G. Barbounis, J. B. Theocharis, M. C. Alexiadis, and P. S. Dokopoulos. Long-term wind speed and power forecasting using local recurrent neural network models. *IEEE Trans. Power Syst.*, 21(1):273–284, 2006.

[10] L. T. Biegler, I. E. Grossmann, and A. W. Westerberg. *Systematic Methods for Chemical Process Design*. Prentice Hall, 1997.

[11] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[12] B. Borchers and J. E. Mitchell. An improved branch and bound algorithm for mixed integer nonlinear programming. *Comput. Oper. Res.*, 21(4):359–367, 1994.

[13] L. M. C. Braz, C. A. Castro, and C. A. F. Murari. A critical evaluation of step size optimization based load flow methods. *IEEE Trans. Power Syst.*, 15(1):202–207, 2000.

[14] G. Brown. *Diversity in Neural Network Ensembles*. PhD thesis, University of Birmingham, Birmingham, United Kingdom, 2003.

[15] H. E. Brown, G. K. Carter, H. H. Happ, and C. E. Person. Power flow solution by impedance matrix iterative method. *IEEE Trans. Power App. Syst.*, 82(65):1–10, 1963.

[16] D. W. Bunn. Forecasting loads and prices in competitive power markets. *Proc. IEEE*, 88(2):163–169, 2000.

[17] M. R. Bussieck. *Optimal lines in public rail transport*. PhD thesis, TU Braunschweig, Brunswick, Germany, 1998.

[18] R. H. Byrd, F. E. Curtis, and J. Nocedal. Infeasibility detection and sqp methods for nonlinear optimization. *SIAM J. Optimiz.*, 20(5):2281–2299, 2010.

[19] R. H. Byrd, M. E. Hribar, and J. Nocedal. An interior point algorithm for large-scale nonlinear programming. *SIAM J. Optimiz.*, 9(4):877–900, 1999.

[20] R. H. Byrd, J. Nocedal, and R. A. Waltz. Feasible interior methods using slacks for nonlinear optimization. *Comput. Optim. Appl.*, 26(1):35–61, 2003.

[21] C. Byrne and Y. Censor. Proximity function minimization using multiple bregman projections, with applications to split feasibility and kullback-leibler distance minimization. *Ann. Oper. Res.*, 105:77–98, 2001.

[22] F. Capitanescu and L. Wehenkel. Sensitivity-based approaches for handling discrete variables in optimal power flow computations. *IEEE Trans. Power Syst.*, 25(4):1780–1789, 2010.

[23] J. Carpentier. Contribution to the economic dispatch problem. *Bull. Soc. France Elect.*, 8:431–437, 1962.

[24] P. A. Castillo, J. J. Merelo, et al. Comparing evolutionary hybrid systems for design and optimization of multilayer perceptron structure along training parameters. *Inform. Sciences*, 177(14):2884–2905, 2007.

[25] S. Chakrabarti, E. Kyriakides, and D. G. Eliades. Placement of synchronized measurements for power system observability. *IEEE Trans. Power Del.*, 24(1):12–19, 2009.

[26] C. C. Chang and C. J. Lin. LIBSVM: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2:27:1–27:27, 2011.

[27] W. Charytoniuk and M. S. Chen. Very short-term load forecasting using artificial neural networks. *IEEE Trans. Power Syst.*, 15(1):263–268, 2000.

[28] W. Charytoniuk, M. S. Chen, and P. Van Olinda. Nonparametric regression based short-term load forecasting. *IEEE Trans. Power Syst.*, 13(3):725–730, 1998.

[29] B. J. Chen, M. W. Chang, and C. J. Lin. Load forecasting using support vector machines: A study on eunite competition 2001. *IEEE Trans. Power Syst.*, 19(4):1821–1830, 2004.

[30] J. H. Chen. *Hybrid Trust-Tech Algorithms and Their Applications to Mixed INTEGER and Mini-Max Optimization Problems*. PhD thesis, Cornell University, Ithaca, NY, 2007.

[31] H. D. Chiang, J. H. Chen, and C. K. Reddy. Trust-tech-based global optimization methodology for nonlinear programming. In *Lectures on Global Optimization*, pages 47–70. American Mathematical Society, 2009.

[32] H. D. Chiang and C. C. Chu. A systematic search method for obtaining multiple local optimal solutions of nonlinear programming problems. *IEEE Trans. Circuits Syst.*, 43(2):99–109, 1996.

[33] H. D. Chiang and A. Fekih-Ahmed. Quasi-stability regions of nonlinear dynamical systems: Theory. *IEEE Trans. Circuits Syst.*, 43(8):627–635, 1996.

[34] H. D. Chiang, A. J. Flueck, K. S. Shah, and N. Balu. CPFLOW: a practical tool for tracing power system steady-state stationary behavior due to load and generation variations. *IEEE Trans. Power Syst.*, 10(2):623–634, 1995.

[35] H. D. Chiang, M. W. Hirsch, and F. F. Wu. Stability region of nonlinear

autonomous dynamical systems. *IEEE Trans. Autom. Control*, 33(1):16–27, 1988.

[36] H. D. Chiang and J. Lee. Trust-tech paradiagm for computing high-quality optimal solutions: Method and theory. In *Modern Heuristic Optimization Techniques*, pages 209–233. IEEE Press, 2008.

[37] H. D. Chiang and C. K. Reddy. TRUST-TECH based neural network training. In *Proc. Int. Joint Conf. Neural Netw.*, pages 90–95, Orlando, FL, August 2007.

[38] H. D. Chiang, B. Wang, and Q. Y. Jiang. Applications of trust-tech methodology in optimal power flow of power systems. In *Optimization in the Energy Industry*, pages 297–318. Springer-Verlag, New York, NY, 2009.

[39] H. D. Chiang, J. C. Wang, O. Cockings, and H. D. Shin. Optimal capacitor placements in distribution systems - i: a new formulation and the overall problem. *IEEE Trans. Power Del.*, 5(2):634–642, 1990.

[40] H. D. Chiang, F. F. Wu, and P. P. Varaiya. Foundations of the potential energy boundary surface method for power system transient stability analysis. *IEEE Trans. Circuits Syst.*, 35(6):712–728, 1988.

[41] H. D. Chiang, F. F. Wu, and P. P. Varaiya. A bcu method for direct analysis of power system transient stability. *IEEE Trans. Power Syst.*, 9(3):1194–1208, 1994.

[42] J. W. Chinneck. The constraint consensus method for finding approximately feasible points in nonlinear programs. *INFORMS J. Comput.*, 16(3):255–265, 2004.

[43] J. W. Chinneck. *Feasibility and Infeasibility in Optimization: Algorithms and Computational Methods*. Springer, 2008.

[44] M. H. Choueiki, C. A. Mount-Compbell, and S. C. Ahalt. Implementing a weighted least squares procedure in training a neural network to solve the short-term load forecasting problem. *IEEE Trans. Power Syst.*, 12(4):1689–1694, 1997.

[45] M. T. Claessens, N. M. van Dijk, and P. J. Zwaneveld. Cost optimal allocation of rail passenger lines. *Eur. J. Oper. Res.*, 110(3):474–489, 1998.

[46] T. S. Coffey, C. T. Kelley, and D. E. Keys. Pseudotransient continuation and differential-algebraic equations. *SIAM J. Sci. Comput.*, 25(2):553–569, 2003.

[47] M. Daneshdoost, M. Lotfalian, G. Bumroonggit, and J. P. Ngoy. Neural network with fuzzy set-based classification for short-term load forecasting. *IEEE Trans. Power Syst.*, 13(4):1386–1391, 1998.

[48] F. de Leon and A. Semlyen. Iterative solvers in the newton power flow problem: preconditioners, inexact solutions and partial jacobian updates. *IEE Proc. Gener. Transm. Distrib.*, 149(4):479–484, 2002.

[49] M. Delgado, M. P. Cuellar, and M. C. Pegalajar. Multiobjective hybrid optimization and training of recurrent neural networks. *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, 35(2):381–385, 2005.

[50] I. Diener. Trajectory methods in global optimization. In *Handbook of Global Optimization*, pages 649–668. Kluwer Academic Publishers, 1996.

[51] H. W. Dommel and W. F. Tinney. Optimal power flow solutions. *IEEE Trans. Power App. Syst.*, 87:1866–1876, 1968.

[52] M. A. Duran and I. E. Grossmann. An outer approximation algorithm for a class of mixed-integer nonlinear programs. *Math. Program.*, 36(3):307–339, 1986.

[53] O. A. Elwakeil and J. S. Arora. Methods for finding feasible points in constrained optimization. *AIAA J.*, 33(9):1715–1719, 1995.

[54] R. E. Fan, P. H. Chen, and C. J. Lin. Working set selection using second order information for training svm. *J. Mach. Learn. Res.*, 6:1889–1918, 2005.

[55] S. Fan and L. N. Chen. Short-term load forecasting based on an adaptive hybrid method. *IEEE Trans. Power Syst.*, 21(1):392–401, 2006.

[56] M. C. Ferris, C. Kanzow, and T. S. Munson. Feasible descent algorithms for mixed complementarity problems. *Math. Program.*, 86(3):475–497, 1998.

[57] C. A. Floudas, P. M. Pardalos, et al. *Handbook of Test Problems in Local and Global Optimization*. Kluwer Academic Publishers, 1999.

[58] C. A. Floudas and G. Paules. A mixed-integer nonlinear programming formulation for the synthesis of heat integrated distillation sequence. *Comput. Chem. Eng.*, 12(6):531–546, 1988.

[59] A. Forsgren and P. E. Gill. Primal-dual interior methods for nonconvex nonlinear programming. *SIAM J. Optimiz.*, 8(4):1132–1152, 1998.

[60] A. S. Fraser. Simulation of genetic systems by automatic digital computers. I. Introduction. *Australian J. Biol. Sci.*, 10:484–491, 1957.

[61] G. Fumera, F. Roli, and A. Serrau. A theoretical analysis of bagging as a linear combination of classifiers. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(7):1293–1299, 2008.

[62] N. Garcia-Pedrajas, C. Garcia-Osorio, and C. Fyfe. Nonlinear boosting projections for ensemble construction. *J. Mach. Learn. Res.*, 8:1–33, 2007.

[63] N. Garcia-Pedrajas, C. Hervas-Martinez, and J. Munoz-Perez. COVNET: a cooperative coevolutionary model for evolving artificial neural networks. *IEEE Trans. Neural Netw.*, 14(3):575–596, 2003.

[64] N. Garcia-Pedrajas, C. Hervas-Martinez, and D. Oriz-Boyer. Cooperative coevolution of artificial neural network ensembles for pattern classification. *IEEE Trans. Evol. Comput.*, 9(3):271–302, 2005.

[65] A. M. Geoffrion. Generalized benders decomposition. *J. Optimiz. Theory App.*, 10(4):237–260, 1972.

[66] P. R. Gill, W. Murray, and M. H. Wright. The levenberg-marquardt method. In *Practical Optimization*, pages 136–137. Academic Press, London, UK, 1981.

[67] J. Gonzalez, I. Rojas, et al. Multiobjective evolutionary optimization of the size, shape, and position parameters of radial basis function networks for function approximation. *IEEE Trans. Neural Netw.*, 14(6):1478–1495, 2003.

[68] E. Gonzalez-Romera, M. A. Jaramillo-Moran, and D. Carmona-Fernandez. Monthly electric energy demand forecasting based on trend extraction. *IEEE Trans. Power Syst.*, 21(4):1946–1953, 2006.

[69] M. Gori and A. Tesi. On the problem of local minima in backpropagation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(1):76–86, 1992.

[70] J. J. Govindhasamy, S. F. McLoone, and G. W. Irwin. Second-order training of adaptive critics for online process control. *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, 35(2):381–385, 2005.

[71] G. Gross and F. D. Galiana. Short-term load forecasting. *Proc. IEEE*, 75(12):1558–1573, 1987.

[72] I. E. Grossmann and Z. Kravanja. Mixed-integer nonlinear programming: a survey of algorithms and applications. In L. T. Biegler, T. F. Coleman, et al., editors, *Large-Scale Optimization with Applications: Part II: Optimal Design and Control*, pages 73–100. Springer Verlag, 1997.

[73] J. Guckenheimer and P. Holmes. *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields*. Springer, 1983.

[74] O. K. Gupta and A. Ravindran. Branch and bound experiments in convex nonlinear integer programming. *Manage. Sci.*, 31(12):1533–1546, 1985.

[75] T. Haida and S. Muto. Regression based peak load forecasting using a transformation technique. *IEEE Trans. Power Syst.*, 9(4):1788–1794, 1994.

[76] S. Hashem. Optimal linear combination of neural networks. *Neural Netw.*, 10(4):599–614, 1997.

[77] S. Haykin. *Neural Networks and Learning Machines*. Prentice Hall, 3 edition, 2009.

[78] H. S. Hippert, C. E. Pedreira, and R. C. Souza. Neural networks for short-term load forecasting: A review and evaluation. *IEEE Trans. Power Syst.*, 16(1):44–55, 2001.

[79] M. W. Hirsch and C. C. Pugh. Stable manifolds and hyperbolic sets. *P. Symp. Pure Math.*, 14:133–163, 1970.

[80] K. L. Ho, Y. Y. Hsu, C. F. Chen, T. E. Lee, C. C. Liang, T. S. Lai, and K. K. Chen. Short term load forecasting of taiwan power system using a knowledge-based expert system. *IEEE Trans. Power Syst.*, 5(4):1214–1221, 1990.

[81] G. B. Huang. Learning capability and storage capacity of two-hidden-layer feedforward networks. *IEEE Trans. Neural Netw.*, 14(2):274–281, 2003.

[82] S. J. Huang and K. R. Shih. Short-term load forecasting via arma model identification including non-gaussian process considerations. *IEEE Trans. Power Syst.*, 18(2):673–679, 2003.

[83] W. Ibrahima and J. W. Chinneck. Improving solver success in reaching feasibility for sets of nonlinear constraints. *Comput. Oper. Res.*, 35(5):1394–1411, 2008.

[84] D. G. Infield and D. C. Hill. Optimal smoothing for trend removal in short term electricity demand forecasting. *IEEE Trans. Power Syst.*, 13(3):1115–1120, 1998.

[85] M. M. Islam, X. Yao, and K. Murase. A constructive algorithm for training cooperative neural network ensembles. *IEEE Trans. Neural Netw.*, 14(4):820–834, 2003.

[86] A. K. Jain, R. P. W. Duin, and J. C. Mao. Statistical pattern recognition: a review. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(1):4–37, 2000.

[87] J. Jardim and B. Stott. Synthetic dynamics power flow. In *Proceedings of IEEE Power Engineering Society General Meeting*, volume 1, pages 479–484, San Mateo, CA, 2005.

[88] O. De Jesus and M. T. Hagan. Backpropagation algorithms for a broad class of dynamic networks. *IEEE Trans. Neural Netw.*, 18(1):14–27, 2007.

[89] H. Th. Jongen, P. Jonker, and F. Twilt. *Nonlinear Optimization in R_n*. Peter Lang Verlag, 1986.

[90] C. F. Juang. A hybrid of genetic algorithm and particle swarm optimization for recurrent network design. *IEEE Trans. Syst., Man, Cybern. B*, 34(2):997–1006, 2004.

[91] N. Karmarkar. A new polynomial time algorithm for linear programming. *Combinatorica*, 4(4):373–395, 1984.

[92] C. R. Karrem. *TRUST-TECH Based Methods for Optimization and Learning*. PhD thesis, Cornell University, Ithaca, NY, 2007.

[93] C. T. Kelley and D. E. Keyes. Convergence analysis of pseudo-transient continuation. *SIAM J. Numer. Anal.*, 35(2):508–523, 1998.

[94] A. Khotanzad, R. Afkhami-Rohani, et al. Annstlf - a neural-network-based electric load forecasting system. *IEEE Trans. Neural Netw.*, 8(4):835–846, 1997.

[95] A. Khotanzad, R. Afkhami-Rohani, and D. Maratukulam. ANNSTLF - artificial neural network short-term load forecaster - generation three. *IEEE Trans. Power Syst.*, 13(4):1413–1422, 1998.

[96] A. Khotanzad, E. W. Zhou, and H. Elragal. A neuro-fuzzy approach to short-term load forecasting in a price-sensitive environment. *IEEE Trans. Power Syst.*, 17(4):1273–1282, 2002.

[97] K. H. Kim, J. K. Park, K. J. Hwang, and S. H. Kim. Implementation of hybrid short-term load forecasting system using artificial neural networks and fuzzy expert systems. *IEEE Trans. Power Syst.*, 10(3):1534–1539, 1995.

[98] A. E. Kostopoulos and T. N. Grapsa. Self-scaled conjugate gradient training algorithms. *Neurocomputing*, 72(13-15):525–533, 2009.

[99] L. I. Kuncheva and C. J. Whitaker. Measures of diversity in classifier ensembles and their relationship with ensemble accuracy. *Mach. Learn.*, 51:181–207, 2003.

[100] D. J. Laughhunn. Quadratic binary programming with applications to capital-budgeting problems. *Oper. Res.*, 18(3):454–461, 1970.

[101] J. Lee. *Trajectory-based methods for global optimization: theory and algorithms*. PhD thesis, Cornell University, Ithaca, NY, 1999.

[102] J. Lee. An optimization-driven framework for the computation of the controlling uep in transient stability analysis. *IEEE Trans. Autom. Control*, 49(1):115–119, 2004.

[103] J. Lee and H. D. Chiang. Theory of stability regions for a class of nonhyperbolic dynamical systems and its application to constraint satisfaction problems. *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, 49(2):196–209, 2002.

[104] J. Lee and H. D. Chiang. A dynamical trajectory-based methodology for systematically computing multiple optimal solutions of general nonlinear programming problems. *IEEE Trans. Autom. Control*, 49(6):888–899, 2004.

[105] J. S. Lee, I. S. Oh, and B. R. Moon. Hybrid genetic algorithms for feature selection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(11):1424–1437, 2004.

[106] K. Y. Lee, Y. T. Chan, and J. H. Park. Short-term load forecasting using an artificial neural network. *IEEE Trans. Power Syst.*, 7(1):124–132, 1992.

[107] T. Leon and V. Liern. A fuzzy method to repair infeasibility in linearly constrained problems. *Fuzzy Set. Syst.*, 122(2):237–243, 2001.

[108] P. Leray and P. Gallinari. Feature selection with neural networks. *Behaviormetrika*, 26(1):145–166, 1999.

[109] F. H. F. Leung, H. K. Lam, S. H. Ling, and P. K. S. Tam. Tuning of the structure and parameters of a neural network using an improved genetic algorithm. *IEEE Trans. Neural Netw.*, 9(9):79–88, 2003.

[110] S. H. Li and H. D. Chiang. Nonlinear predictors and hybrid corrector for fast continuation power flow. *IET Gener. Transm. Distrib.*, 2(3):341–354, 2008.

[111] T. Li and M. Shahidehpour. Price-based unit commitment: a case of lagrangian relaxation versus mixed integer programming. *IEEE Trans. Power Syst.*, 20(4):2015–2025, 2005.

[112] G. C. Liao and T. P. Tsao. Application of a fuzzy neural network combined with a chaos genetic algorithm and simulated annealing to short-term load forecasting. *IEEE Trans. Evol. Comput.*, 10(3):330–340, 2006.

[113] H. T. Lin and L. Li. Support vector machinery for infinite ensemble learning. *J. Mach. Learn. Res.*, 9:285–312, 2008.

[114] D. R. Liu, T. S. Chang, and Y. Zhang. A constructive algorithm for feedforward neural networks with incremental training. *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, 49(12):1876–1879, 2002.

[115] H. F. Liu, L. C. Jin, J. D. McCalley, R. Kumar, V. Ajjarapu, and N. Elia. Planning reconfigurable reactive control for voltage stability limited power systems. *IEEE Trans. Power Syst.*, 24(2):1029–1038, 2009.

[116] J. Liu, W. C. Zhong, and L. C. Jiao. A multiagent evolutionary algorithm for constraint satisfaction problems. *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, 35(5):54–73, 2006.

[117] C. N. Lu, H. T. Wu, and S. Vemuri. Neural network based short term load forecasting. *IEEE Trans. Power Syst.*, 8(1):336–342, 1993.

[118] T. B. Ludermir, A. Yamazaki, and C. Zanchettin. An optimization methodology for neural network weights and architectures. *IEEE Trans. Neural Netw.*, 17(6):1452–1459, 2006.

[119] L. Luksan, C. Matonoha, and J. Vlcek. Interior-point method for nonlinear non-convex optimization. *Numer. Linear Algebr.*, 11(5-6):431–453, 2004.

[120] J. C. T. Mao and B. A. Wallingford. An extension of lawler and bell's method of discrete optimization with examples from capital budgeting. *Manage. Sci.*, 15(2):51–60, 1968.

[121] S. Mehrotra. On the implementation of a primal-dual interior point method. *SIAM J. Optimiz.*, 2(4):575–601, 1992.

[122] J. Meller, M. Wagner, and R. Elber. Maximum feasibility guideline to the design and analysis of protein folding potentials. *J. Comput. Chem.*, 23(1):111–118, 2002.

[123] M. Milano, P. Koumoutsakos, and J. Schmidhuber. Self-organizing nets for optimization. *IEEE Trans. Neural Netw.*, 15(3):758–765, 2004.

[124] O. Mohammed, D. Park, R. Merchant, et al. Practical experiences with an adaptive neural network short-term load forecasting system. *IEEE Trans. Power Syst.*, 10(1):254–265, 1995.

[125] J. A. Momoh, M. E. El-Hawary, and R. Adapa. A review of selected optimal power flow literature to 1993: Part-I and part-II. *IEEE Trans. Power Syst.*, 14:96–111, 1999.

[126] A. Monticelli, A. Garcia, and O. R. Saavedra. Fast decoupled load flow: Hypothesis, derivations, and testing. *IEEE Trans. Power Syst.*, 5(4):1425–1431, 1990.

[127] H. Mori and H. Kobayashi. Optimal fuzzy inference for short-term load forecasting. *IEEE Trans. Power Syst.*, 11(1):390–396, 1996.

[128] U. Naftaly, U. Intrator, and D. Horn. Optimal ensemble averaging of neural networks. *Network*, 8(3):283–296, 1997.

[129] J. Nocedal and S. J. Wright. *Numer. Optimiz.* Springer, 2 edition, 2006.

[130] D. Opitz and R. Maclin. Popular ensemble methods: An empirical study. *J. Artif. Intell. Res.*, 11:169–198, 1999.

[131] D. W. Opitz and J. W. Shavlik. Generating accurate and diverse members of a neural-network ensemble. In *Adv. Neural Inf. Proc. Syst.*, volume 8, pages 535–541, Denver, CO, 1996.

[132] N. P. Padhy. Unit commitment-a bibliographical survey. *IEEE Trans. Power Syst.*, 19(2):1196–1205, 2004.

[133] A. Pages, J. Gondzio, and N. Nabona. Warmstarting for interior point methods applied to the long-term power planning problem. *Eur. J. Oper. Res.*, 197(1):112–125, 2009.

[134] P. P. Palmes, T. Hayasaka, and S. Usui. Mutation-based genetic neural network. *IEEE Trans. Neural Netw.*, 16(3):587–600, 2005.

[135] A. D. Papalexopoulos, S. Y. Hao, and T. M. Peng. An implementation of a neural network based load forecasting model for the ems. *IEEE Trans. Power Syst.*, 9(4):1956–1961, 1994.

[136] D. C. Park, M. A. El-Sharkawi, et al. Electric load forecasting using an artificial neural network. *IEEE Trans. Power Syst.*, 6(2):442–449, 1991.

[137] J. H. Park, Y. M. Park, and K. Y. Lee. Composite modeling for adaptive short-term load forecasting. *IEEE Trans. Power Syst.*, 6(2):450–457, 1991.

[138] T. M. Peng, N. F. Hubele, and G. G. Karady. An adaptive neural iwlwork approach to one-week ahead load forecasting. *IEEE Trans. Power Syst.*, 8(3):1195–1203, 1993.

[139] F. A. Potra and S. J. Wright. Interior-point methods. *J. Comput. Appl. Math.*, 124(1-2):281–302, 2000.

[140] S. Rahman and O. Hazim. A generalized knowledge-based short-term load-forecasting technique. *IEEE Trans. Power Syst.*, 8(2):508–514, 1993.

[141] D. K. Ranaweera, G. G. Karady, and R. G. Farmer. Economic impact analysis of load forecasting. *IEEE Trans. Power Syst.*, 12(3):1388–1392, 1997.

[142] N. S. V. Rao. On fusers that perform better than best sensor. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(8):904–909, 2001.

[143] C. K. Reddy, H. D. Chiang, and B. Rajaratnam. Trust-tech based expectation maximization fro learning finite mixture models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(7):1146–1157, 2008.

[144] A. L. Rendell and R. Sheshu. Learning hard concepts through constructive induction: framework and rationale. *Comput. Intell.*, 6(4):247–270, 1990.

[145] J. Renegar. *A Mathematical View of Interior-Point Methods in Convex Optimization*. SIAM, 2001.

[146] B. D. Ripley and N. L. Hjort. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1995.

[147] I. Roytelman, B. K. Wee, and R. L. Lugtu. Volt/var control algorithm for modern distribution management system. *IEEE Trans. Power Syst.*, 10(3):1454–1460, 1995.

[148] N. V. Sahinidis and M. Tawarmalani. Applications of global optimization to process and molecular design. *Comput. Chem. Eng.*, 24(9):2157–2169, 2000.

[149] A. Sard. The measure of the critical values of differentiable maps. *Bull. Amer. Math. Soc.*, 48(12):883–890, 1942.

[150] A. M. Sasson, C. Trevino, and F. Aboytes. Improved newton's load flow through a minimization technique. *IEEE Trans. Power App. Syst.*, 90(5):116–122, 1971.

[151] T. Senjyu, P. Mandal, K. Uezato, and T. Funabashi. Next day load curve forecasting using hybrid correction method. *IEEE Trans. Power Syst.*, 20(1):102–109, 2005.

[152] T. Senjyu, K. Shimabukuro, K. Uezato, and T. Funabashi. A fast technique for unit commitment problem by extended priority list. *IEEE Trans. Power Syst.*, 18(2):882–888, 2003.

[153] T. Senjyu, H. Takara, K. Uezato, and T. Funabashi. One-hour-ahead load forecasting using neural network. *IEEE Trans. Power Syst.*, 17(1):113–118, 2002.

[154] Y. Shang and B. W. Wah. Global optimization for neural network training. *IEEE Computer*, 29(3):45–54, 1996.

[155] A. Sharkey. On combining artificial neural nets. *Connect. Sci.*, 8(3-4):299–314, 1996.

[156] K. Socha and C. Blum. An ant colony optimization algorithm for continuous optimization: application to feed-forward neural network training. *Neural Comput. Appl.*, 16(3):235–247, 2007.

[157] S. Sohn and C. H. Dagli. Ensemble of evolving neural networks in classification. *Neural Process. Lett.*, 19(3):191–203, 2004.

[158] K. B. Song, Y. S. Baek, D. H. Hong, and G. Jang. Short-term load forecasting for the holidays using fuzzy linear regression method. *IEEE Trans. Power Syst.*, 20(1):96–101, 2005.

[159] S. N. Talukdar and F. F. Wu. Computer-aided dispatch for electric power systems. *Proc. IEEE*, 69:1212–1231, 1981.

[160] R. Tanabe, K. Yasuda, R. Yokoyama, and H. Sasaki. Flexible generation mix under multiobjectives and uncertainties. *IEEE Trans. Power Syst.*, 8(2):581–587, 1993.

[161] M. Tawarmalani and N. Sahinidis. Exact algorithms for global optimization of mixed-integer nonlinear programs. In P. M. Pardalos and E. Romeijn, editors, *Handbook of Global Optimization*, pages 65–85. Kluwer Academic Publishers, 2001.

[162] M. Tawarmalani and N. Sahinidis. *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithm, Software and Applications*. Springer, 2002.

[163] W. F. Tinney and C. E. Hart. Power flow solution by newton's method. *IEEE Trans. Power App. Syst.*, 86(11):1449–1460, 1967.

[164] A. L. Tits, A. Wachter, S. Bakhtiari, T. J. Urban, and C. T. Lawrence. A primal-dual interior-point method for nonlinear programming with strong global and local convergence properties. *SIAM J. Optimiz.*, 14(1):173–199, 2003.

[165] N. Ueda. Optimal linear combination of neural networks for improving classification performance. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(2):207–215, 2000.

[166] J. Vermaak and E. C. Botha. Recurrent neural networks for short-term load forecasting. *IEEE Trans. Power Syst.*, 13(1):126–132, 1998.

[167] A. Wachter and L. T. Biegler. Failure of global convergence for a class of interior point methods for nonlinear programming. *Math. Program.*, 88(3):565–574, 2000.

[168] A. Wachter and L. T. Biegler. Line search filter methods for nonlinear programming: Motivation and global convergence. *SIAM J. Optimiz.*, 16(1):1–31, 2005.

[169] A. Wachter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Math. Program.*, 106(1):25–57, 2006.

[170] A. Wachter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. In *Mathematical Programming 106*, pages 25–57, 2006.

[171] B. Wang and H. D. Chiang. Elite: Ensemble of optimal input-pruned neural networks using trust-tech. *IEEE Trans. Neural Netw.*, 22(1):96–109, 2011.

[172] T. Westerlund, H. Skrifvars, I. Harjunkoski, and R. Porn. An extended cutting plane method for a class of non-convex minlp problems. *Comput. Chem. Eng.*, 22(3):357–365, 1998.

[173] T. Windeatt. Accuracy/diversity and ensemble MLP classifier design. *IEEE Trans. Neural Netw.*, 17(5):1194–1211, 2006.

[174] M. H. Wright. The interior-point revolution in optimization: history, recent developments, and lasting consequences. *Bull. Amer. Math. Soc.*, 42(1):39–56, 2005.

[175] S. J. Wright. *Primal-Dual Interior-Point Methods*. SIAM, 1997.

[176] Z. L. Xu, I. King, M. R. T. Lyu, and R. Jin. Discriminative semi-supervised feature selection via manifold regularization. *IEEE Trans. Neural Netw.*, 21(7):1033–1047, 2010.

[177] H. Yamashita. A differential equation approach to nonlinear programming. *Math. Program.*, 18(1):155–168, 1980.

[178] C. M. Yang and J. L. Beck. Generalized trajectory methods for finding multiple extrema and roots of functions. *J. Optimiz. Theory App.*, 97(1):211–227, 1998.

[179] H. T. Yang, C. M. Huang, and C. L. Huang. Identification of armax model for short term load forecasting: an evolutionary programming approach. *IEEE Trans. Power Syst.*, 11(1):403–408, 1996.

[180] J. Yang. Infeasibility resolution based on goal programming. *Comput. Oper. Res.*, 35(5):1483–1493, 2008.

[181] X. Yao. Evolving artificial neural networks. *Proc. IEEE*, 87(9):1423–1447, 1999.

[182] X. Yao and Y. Liu. A new evolutionary system for evolving artificial neural networks. *IEEE Trans. Neural Netw.*, 8(3):694–713, 1997.

[183] X. Yao and Y. Liu. Making use of population information in evolutionary artificial neural networks. *IEEE Trans. Syst., Man, Cybern., B*, 28(3):417–425, 1998.

[184] J. Ye, J. F. Qiao, et al. A Tabu based neural network learning algorithm. *Neurocomputing*, 70(4-6):875–882, 2007.

[185] H. Yoo and R. L. Pimmel. Short term load forecasting using a self-supervised adaptive neural network. *IEEE Trans. Power Syst.*, 14(2):779–784, 1999.

[186] H. Yoshida, K. Kawata, Y. Fukuyama, S. Takayama, and Y. Nakanishi. A particle swarm optimization for reactive power and voltage control considering voltage security assessment. *IEEE Trans. Power Syst.*, 15(4):1232–1239, 2000.

[187] Z. W. Yu. A temperature match based opttmizatiion method for daily load prediction considering dlc effect. *IEEE Trans. Power Syst.*, 11(2):728–733, 1996.

[188] G. Zhang, B. E. Patuwo, and M.Y. Hu. Forecasting with artificial neural networks: The state of the art. *Int. J. Forecast.*, 14(1):35–62, 1998.

[189] W. J. Zhang, F. X. Li, and L. M. Tolbert. Review of reactive power planning: objectives, constraints, and algorithms. *IEEE Trans. Power Syst.*, 22(4):2177–2186, 2007.

[190] Z. H. Zhou, J. X. Wu, and W. Tang. Ensembling neural networks: many could be better than all. *Artif. Intell.*, 137(1-2):239–263, 2002.